SCC.C - Linux driver for Z8530 based HDLC cards for AX.25

This is a subset of the documentation. To use this driver you MUST have the full package from:

Internet:

1. ftp://ftp.ccac.rwth-aachen.de/pub/jr/z8530drv-utils_3.0-3.tar.gz
2. ftp://ftp.pspt.fi/pub/ham/linux/ax25/z8530drv-utils_3.0-3.tar.gz

Please note that the information in this document may be hopelessly outdated. A new version of the documentation, along with links to other important Linux Kernel AX.25 documentation and programs, is available on http://yaina.de/jreuter

1. Initialization of the driver

To use the driver, 3 steps must be performed:

1. if compiled as module: loading the module
2. Setup of hardware, MODEM and KISS parameters with sccinit
3. Attach each channel to the Linux kernel AX.25 with "ifconfig"

Unlike the versions below 2.4 this driver is a real network device driver. If you want to run xNOS instead of our fine kernel AX.25 use a 2.x version (available from above sites) or read the AX.25-HOWTO on how to emulate a KISS TNC on network device drivers.

1.1 Loading the module

(If you're going to compile the driver as a part of the kernel image,
skip this chapter and continue with 1.2)

Before you can use a module, you'll have to load it with:

```
insmod scc.o
```

please read 'man insmod' that comes with module-init-tools.

You should include the insmod in one of the /etc/rc.d/rc.* files, and don't forget to insert a call of sccinit after that. It will read your /etc/z8530drv.conf.

1.2. /etc/z8530drv.conf

To setup all parameters you must run /sbin/sccinit from one of your rc.*-files. This has to be done BEFORE you can "ifconfig" an interface. Sccinit reads the file /etc/z8530drv.conf and sets the hardware, MODEM and KISS parameters. A sample file is delivered with this package. Change it to your needs.

The file itself consists of two main sections.

1.2.1 configuration of hardware parameters

The hardware setup section defines the following parameters for each Z8530:

```
chip    1
data_a  0x300                   # data port A
ctrl_a  0x304                   # control port A
data_b  0x301                   # data port B
ctrl_b  0x305                   # control port B
irq     5                       # IRQ No. 5
pclock  4915200                 # clock
board   BAYCOM                  # hardware type
escc    no                      # enhanced SCC chip? (8580/85180/85280)
vector  0                       # latch for interrupt vector
special no                      # address of special function register
option  0                       # option to set via sfr
```

chip

- this is just a delimiter to make sccinit a bit simpler to program. A parameter has no effect.

data_a

- the address of the data port A of this Z8530 (needed)

ctrl_a

- the address of the control port A (needed)

data_b

- the address of the data port B (needed)

ctrl_b

- the address of the control port B (needed)

irq

- the used IRQ for this chip. Different chips can use different IRQs or the same. If they share an interrupt, it needs to be specified within one chip-definition only.

pclock - the clock at the PCLK pin of the Z8530 (option, 4915200 is

default), measured in Hertz

board

- the "type" of the board:

SCC type	**value**
PA0HZP SCC card	PA0HZP
EAGLE card	EAGLE
PC100 card	PC100
PRIMUS-PC (DG9BL) card	PRIMUS
BayCom (U)SCC card	BAYCOM

escc

- if you want support for ESCC chips (8580, 85180, 85280), set this to "yes" (option, defaults to "no")

vector

- address of the vector latch (aka "intack port") for PA0HZP cards. There can be only one vector latch for all chips! (option, defaults to 0)

special

- address of the special function register on several cards. (option, defaults to 0)

option - The value you write into that register (option, default is 0)

You can specify up to four chips (8 channels). If this is not enough, just change:

```
#define MAXSCC 4
```

to a higher value.

Example for the BAYCOM USCC:

```
chip    1
data_a  0x300                   # data port A
ctrl_a  0x304                   # control port A
data_b  0x301                   # data port B
ctrl_b  0x305                   # control port B
irq     5                       # IRQ No. 5 (#)
board   BAYCOM                  # hardware type (*)
#
# SCC chip 2
#
chip    2
data_a  0x302
ctrl_a  0x306
data_b  0x303
ctrl_b  0x307
board   BAYCOM
```

An example for a PA0HZP card:

```
chip 1
data_a 0x153
data_b 0x151
```

```
ctrl_a 0x152
ctrl_b 0x150
irq 9
pclock 4915200
board PA0HZP
vector 0x168
escc no
#
#
#
chip 2
data_a 0x157
data_b 0x155
ctrl_a 0x156
ctrl_b 0x154
irq 9
pclock 4915200
board PA0HZP
vector 0x168
escc no
```

A DRSI would should probably work with this:

(actually: two DRSI cards...)

```
chip 1
data_a 0x303
data_b 0x301
ctrl_a 0x302
ctrl_b 0x300
irq 7
pclock 4915200
board DRSI
escc no
#
#
#
chip 2
data_a 0x313
data_b 0x311
ctrl_a 0x312
ctrl_b 0x310
irq 7
pclock 4915200
board DRSI
escc no
```

Note that you cannot use the on-board baudrate generator off DRSI cards. Use "mode dpll" for clock source (see below).

This is based on information provided by Mike Bilow (and verified by Paul Helay)

The utility "gencfg"

If you only know the parameters for the PE1CHL driver for DOS, run gencfg. It will generate the correct port addresses (I hope). Its parameters are exactly the same as the ones you use with the "attach scc" command in net, except that the string "init" must not appear. Example:

```
gencfg 2 0x150 4 2 0 1 0x168 9 4915200
```

will print a skeleton z8530drv.conf for the OptoSCC to stdout.

```
gencfg 2 0x300 2 4 5 -4 0 7 4915200 0x10
```

does the same for the BAYCOM USCC card. In my opinion it is much easier to edit scc_config.h...

1.2.2 channel configuration

The channel definition is divided into three sub sections for each channel:

An example for scc0:

```
# DEVICE

device scc0     # the device for the following params

# MODEM / BUFFERS

speed 1200              # the default baudrate
clock dpll              # clock source:
                        #       dpll     = normal half duplex operation
```

```
                        #       external = MODEM provides own Rx/Tx clock
                        #       divider  = use full duplex divider if
                        #                  installed (1)
mode nrzi               # HDLC encoding mode
                        #       nrzi = 1k2 MODEM, G3RUH 9k6 MODEM
                        #       nrz  = DF9IC 9k6 MODEM
                        #
bufsize 384             # size of buffers. Note that this must include
                        # the AX.25 header, not only the data field!
                        # (optional, defaults to 384)

# KISS (Layer 1)

txdelay 36              # (see chapter 1.4)
persist 64
slot    8
tail    8
fulldup 0
wait    12
min     3
maxkey  7
idle    3
maxdef  120
group   0
txoff   off
softdcd on
slip    off
```

The order WITHIN these sections is unimportant. The order OF these sections IS important. The MODEM parameters are set with the first recognized KISS parameter...

Please note that you can initialize the board only once after boot (or insmod). You can change all parameters but "mode" and "clock" later with the Sccparam program or through KISS. Just to avoid security holes...

1. this divider is usually mounted on the SCC-PBC (PA0HZP) or not present at all (BayCom). It feeds back the output of the DPLL (digital pll) as transmit clock. Using this mode without a divider installed will normally result in keying the transceiver until maxkey expires --- of course without sending anything (useful).

2. Attachment of a channel by your AX.25 software

2.1 Kernel AX.25

To set up an AX.25 device you can simply type:

```
ifconfig scc0 44.128.1.1 hw ax25 dl0tha-7
```

This will create a network interface with the IP number 44.128.20.107 and the callsign "dl0tha". If you do not have any IP number (yet) you can use any of the 44.128.0.0 network. Note that you do not need axattach. The purpose of axattach (like slattach) is to create a KISS network device linked to a TTY. Please read the documentation of the ax25-utils and the AX.25-HOWTO to learn how to set the parameters of the kernel AX.25.

2.2 NOS, NET and TFKISS

Since the TTY driver (aka KISS TNC emulation) is gone you need to emulate the old behaviour. The cost of using these programs is that you probably need to compile the kernel AX.25, regardless of whether you actually use it or not. First setup your /etc/ax25/axports, for example:

```
9k6     dl0tha-9  9600  255 4 9600 baud port (scc3)
axlink  dl0tha-15 38400 255 4 Link to NOS
```

Now "ifconfig" the scc device:

```
ifconfig scc3 44.128.1.1 hw ax25 dl0tha-9
```

You can now axattach a pseudo-TTY:

```
axattach /dev/ptys0 axlink
```

and start your NOS and attach /dev/ptys0 there. The problem is that NOS is reachable only via digipeating through the kernel AX.25 (disastrous on a DAMA controlled channel). To solve this problem, configure "rxecho" to echo the incoming frames from "9k6" to "axlink" and outgoing frames from "axlink" to "9k6" and start:

```
rxecho
```

Or simply use "kissbridge" coming with z8530drv-utils:

```
ifconfig scc3 hw ax25 dl0tha-9
kissbridge scc3 /dev/ptys0
```

3. Adjustment and Display of parameters

3.1 Displaying SCC Parameters:

Once a SCC channel has been attached, the parameter settings and some statistic information can be shown using the param program:

```
dl1bke-u:~$ sccstat scc0

Parameters:

speed       : 1200 baud
txdelay     : 36
persist     : 255
slottime    : 0
txtail      : 8
fulldup     : 1
waittime    : 12
mintime     : 3 sec
maxkeyup    : 7 sec
idletime    : 3 sec
maxdefer    : 120 sec
group       : 0x00
txoff       : off
softdcd     : on
SLIP        : off

Status:

HDLC                  Z8530           Interrupts         Buffers
-----------------------------------------------------------------------
Sent       :     273  RxOver :     0  RxInts :   125074  Size    :  384
Received   :    1095  TxUnder:     0  TxInts :     4684  NoSpace :    0
RxErrors   :    1591                   ExInts :    11776
TxErrors   :       0                   SpInts :     1503
Tx State   :    idle
```

The status info shown is:

Sent	number of frames transmitted
Received	number of frames received
RxErrors	number of receive errors (CRC, ABORT)
TxErrors	number of discarded Tx frames (due to various reasons)
Tx State	status of the Tx interrupt handler: idle/busy/active/tail (2)
RxOver	number of receiver overruns
TxUnder	number of transmitter underruns
RxInts	number of receiver interrupts
TxInts	number of transmitter interrupts
EpInts	number of receiver special condition interrupts
SpInts	number of external/status interrupts
Size	maximum size of an AX.25 frame (*with* AX.25 headers!)
NoSpace	number of times a buffer could not get allocated

An overrun is abnormal. If lots of these occur, the product of baudrate and number of interfaces is too high for the processing power of your computer. NoSpace errors are unlikely to be caused by the driver or the kernel AX.25.

3.2 Setting Parameters

The setting of parameters of the emulated KISS TNC is done in the same way in the SCC driver. You can change parameters by using the kissparms program from the ax25-utils package or use the program "sccparam":

```
sccparam <device> <paramname> <decimal-|hexadecimal value>
```

You can change the following parameters:

param	**value**
speed	1200
txdelay	36
persist	255

param	value
slottime	0
txtail	8
fulldup	1
waittime	12
mintime	3
maxkeyup	7
idletime	3
maxdefer	120
group	0x00
txoff	off
softdcd	on
SLIP	off

The parameters have the following meaning:

speed:

The baudrate on this channel in bits/sec

Example: sccparam /dev/scc3 speed 9600

txdelay:

The delay (in units of 10 ms) after keying of the transmitter, until the first byte is sent. This is usually called "TXDELAY" in a TNC. When 0 is specified, the driver will just wait until the CTS signal is asserted. This assumes the presence of a timer or other circuitry in the MODEM and/or transmitter, that asserts CTS when the transmitter is ready for data. A normal value of this parameter is 30-36.

Example: sccparam /dev/scc0 txd 20

persist:

This is the probability that the transmitter will be keyed when the channel is found to be free. It is a value from 0 to 255, and the probability is (value+1)/256. The value should be somewhere near 50-60, and should be lowered when the channel is used more heavily.

Example: sccparam /dev/scc2 persist 20

slottime:

This is the time between samples of the channel. It is expressed in units of 10 ms. About 200-300 ms (value 20-30) seems to be a good value.

Example: sccparam /dev/scc0 slot 20

tail:

The time the transmitter will remain keyed after the last byte of a packet has been transferred to the SCC. This is necessary because the CRC and a flag still have to leave the SCC before the transmitter is keyed down. The value depends on the baudrate selected. A few character times should be sufficient, e.g. 40ms at 1200 baud. (value 4) The value of this parameter is in 10 ms units.

Example: sccparam /dev/scc2 4

full:

The full-duplex mode switch. This can be one of the following values:

0: The interface will operate in CSMA mode (the normal
half-duplex packet radio operation)
1: Fullduplex mode, i.e. the transmitter will be keyed at
any time, without checking the received carrier. It will be unkeyed when there are no packets to be sent.
2: Like 1, but the transmitter will remain keyed, also
when there are no packets to be sent. Flags will be sent in that case, until a timeout (parameter 10) occurs.

Example: sccparam /dev/scc0 fulldup off

wait:

The initial waittime before any transmit attempt, after the frame has been queue for transmit. This is the length of the first slot in CSMA mode. In full duplex modes it is set to 0 for maximum performance. The value of this parameter is in 10 ms units.

Example: sccparam /dev/scc1 wait 4

maxkey:

The maximal time the transmitter will be keyed to send packets, in seconds. This can be useful on busy CSMA channels, to avoid "getting a bad reputation" when you are generating a lot of traffic. After the specified time has elapsed, no new frame

will be started. Instead, the trans- mitter will be switched off for a specified time (parameter min), and then the selected algorithm for keyup will be started again. The value 0 as well as "off" will disable this feature, and allow infinite transmission time.

Example: sccparam /dev/scc0 maxk 20

min:

This is the time the transmitter will be switched off when the maximum transmission time is exceeded.

Example: sccparam /dev/scc3 min 10

idle:

This parameter specifies the maximum idle time in full duplex 2 mode, in seconds. When no frames have been sent for this time, the transmitter will be keyed down. A value of 0 is has same result as the fullduplex mode 1. This parameter can be disabled.

Example: sccparam /dev/scc2 idle off # transmit forever

maxdefer

This is the maximum time (in seconds) to wait for a free channel to send. When this timer expires the transmitter will be keyed IMMEDIATELY. If you love to get trouble with other users you should set this to a very low value ;-)

Example: sccparam /dev/scc0 maxdefer 240 # 2 minutes

txoff:

When this parameter has the value 0, the transmission of packets is enable. Otherwise it is disabled.

Example: sccparam /dev/scc2 txoff on

group:

It is possible to build special radio equipment to use more than one frequency on the same band, e.g. using several receivers and only one transmitter that can be switched between frequencies. Also, you can connect several radios that are active on the same band. In these cases, it is not possible, or not a good idea, to transmit on more than one frequency. The SCC driver provides a method to lock transmitters on different interfaces, using the "param <interface> group <x>" command. This will only work when you are using CSMA mode (parameter full = 0).

The number <x> must be 0 if you want no group restrictions, and can be computed as follows to create restricted groups: <x> is the sum of some OCTAL numbers:

200	This transmitter will only be keyed when all other transmitters in the group are off.
100	This transmitter will only be keyed when the carrier detect of all other interfaces in the group is off.
0xx	A byte that can be used to define different groups. Interfaces are in the same group, when the logical AND between their xx values is nonzero.

Examples:

When 2 interfaces use group 201, their transmitters will never be keyed at the same time.

When 2 interfaces use group 101, the transmitters will only key when both channels are clear at the same time. When group 301, the transmitters will not be keyed at the same time.

Don't forget to convert the octal numbers into decimal before you set the parameter.

Example: (to be written)

softdcd:

use a software dcd instead of the real one... Useful for a very slow squelch.

Example: sccparam /dev/scc0 soft on

4. Problems

If you have tx-problems with your BayCom USCC card please check the manufacturer of the 8530. SGS chips have a slightly different timing. Try Zilog... A solution is to write to register 8 instead to the data port, but this won't work with the ESCC chips. *SIGH!*

A very common problem is that the PTT locks until the maxkeyup timer expires, although interrupts and clock source are correct. In most cases compiling the driver with CONFIG_SCC_DELAY (set with make config) solves the problems. For more hints read the (pseudo) FAQ and the documentation coming with z8530drv-utils.

I got reports that the driver has problems on some 386-based systems. (i.e. Amstrad) Those systems have a bogus AT bus timing which will lead to delayed answers on interrupts. You can recognize these problems by looking at the output of Sccstat for the suspected port. If it shows under- and overruns you own such a system.

Delayed processing of received data: This depends on

- the kernel version
- kernel profiling compiled or not
- a high interrupt load
- a high load of the machine --- running X, Xmorph, XV and Povray, while compiling the kernel... hmm ... even with 32 MB RAM ... ;-) Or running a named for the whole .ampr.org domain on an 8 MB box...
- using information from rxecho or kissbridge.

Kernel panics: please read /linux/README and find out if it really occurred within the scc driver.

If you cannot solve a problem, send me

- a description of the problem,
- information on your hardware (computer system, scc board, modem)
- your kernel version
- the output of cat /proc/net/z8530

4. Thor RLC100

Mysteriously this board seems not to work with the driver. Anyone got it up-and-running?

Many thanks to Linus Torvalds and Alan Cox for including the driver in the Linux standard distribution and their support.


```
Joerg Reuter    ampr-net: dl1bke@db0pra.ampr.org
                AX-25   : DL1BKE @ DB0ABH.#BAY.DEU.EU
                Internet: jreuter@yaina.de
                WWW     : http://yaina.de/jreuter
```