HFC-4S / HFC-8S / HFC-E1 Driver Documentation

1. What is what?

The HFC-4S is a single chip controller with 4 S/T interfaces (Basic Rate Interface = BRI). Each interface has two B-channels.

The HFC-8S is a single chip controller with 8 S/T interfaces (Basic Rate Interface = BRI). Each interface has two B-channels.

The HFC-E1 is a single chip controller with one E1 interface (Primary Rate Interface = PRI). The interface has 30 B-channels (by default).

These controllers are designed and produced by Cologne Chip (http://www.colognechip.de).

All of these controllers provide the following features with this driver:

- TE-Mode (connecting a port with an ISDN line)
- NT-Mode (connecting telephones and other terminals to the port)
- Hardware supported DTMF detection (for each B-channel)
- Cross connecting of all B-channels on one chip.
- Cross connecting between chips using PCM bus. (up to 128 calls)
- Conference (up to 8 conferences between all B-channels on one chip)
- *TBD* Simple tone generation (playing tones/patterns with minimum CPU usage)
- Silence generation (playing silence (0-level) without CPU usage)

All these controllers have PCM interface to interconnect channels on one controller and between different controllers. The driver will auto-detect, if cards are connected or not. If multiple cards are connected, the first card found will be PCM master and provides clock to all other cards (slave) connected to it.

Therefore it is essential to sync the first card to your telephone line. If you have multiple lines from your provider, choose any line, because they have all same sync. If you have a PRI line, you should use a PRI interface as PCM master, because this cards gets clock from PRI line and forwards it to PCM interface. If you have BRI line only, you should use card with BRI interface as PCM master. This card automatically detects sync from line and gets the clock from it.

Whenever an interface is in TE-mode, the card retrieves it's clock from it. All interfaces in NT-mode will use the clock and provide it to the connected ISDN device. If no interface is connected to any line (or no TE-mode is used), the card will, if it is PCM master, clock from accurate quartz clock.

2. Loading kernel module

Note: The "mISDN" driver must be compiled and loaded. Follow the instruction on how to load them in the documentation at www.mISDN.org.

```
modprobe hfcmulti
```

will load the card driver with default values for all controllers found. Use 'dmesg' to see all cards found.

```
modprobe hfcmulti \
        [type=<card 1>[,<card 2>[, …]]] \
        [port=<interface 1>[,<interface 2>[, …]]] \
        [pcm=<pcm id 1>[,<pcm id 2>[, …]]] \
        [dslot=<slots 1>[,<slots 2>[, …]]] \
        [iomode=<mode 1>[,<mode 2>[, …]]] \
        [poll = 8 | 16 | 32 | 64 | 128 | 256] \
        [clockdelay_nt] \
        [clockdelay_te] \
        [debug = <flags>]
```

The "**type**" is used to provide feature flags to the cards you have installed in your system. If you have multiple cards, you must given multiple type values. They must be given in same order as PCI subsystem will find them. If you have multiple cards of same type, the first given parameter will be used for the first card found of this type. To see how PCI subsystem will find cards, load hfcmulti without parameters and look at 'dmesg'. The following types are expected:

- Bits 0-7 0x00004 = HFC-4S (4 BRI interfaces)
- Bits 0-7 0x00008 = HFC-8S (8 BRI interfaces)
- Bits 0-7 0x00001 = HFC-E1 (1 PRI interface)

Optional bits can be given with the "type" value. All bits given must be combined.:

- Bit 8 0x00100 = uLaw (instead of aLaw)

Since hardware audio processing is done inside controller core, it must know what audio encoding is used. Change it if you have µ-Law instead of a-Law. (e. g. USA)

- Bit 9 0x00200 = Disable DTMF detection on all B-channels via hardware

Use it to disable hardware DTMF detection. The driver will not use CPU cycles to read DTMF coefficients from controller.

- Bit 10 spare
- Bit 11 0x00800 = Force PCM bus into slave mode. (otherwise auto)
- Bit 12 0x01000 = Force PCM bus into master mode. (otherwise auto)

The first card found will be master automatically. All other cards, if connected via PCM cable, will be slave automatically. Set one of the two bit to force slave or master.

- Bit 13 spare
- Bit 14 0x04000 = Use external ram (128K)
- Bit 15 0x08000 = Use external ram (512K)

By default, the controller has 32K ram for all FIFOs. With external SRAM it can be increased up to almos 8K per FIFO. Set one of these bits, if your card features external SRAM.

- Bit 16 0x10000 = Use 64 timeslots instead of 32
- Bit 17 0x20000 = Use 128 timeslots instead of anything else

By default, the PCM bus has 32 timeslots. The number of PCM timeslots must be equal all cards, that are interconnected. Each slot will be one call, so 64 B-channels can be interconnected with 32 timeslots. For more, set one of these bits.

- Bit 19 0x80000 = Send the Watchdog a Signal (Dual E1 with Watchdog)

Enable watchdog on dual E1 boards. (Detailed description TDB)

(all other bits are reserved and shall be 0)

The "**port**" is used to change port specific parameters. The given number of ports depend on the controller type(s) you use. Example: "type=8,1" will need 9 port parameters to be given. The "port" specifies a bitmap with the following bits:

HFC-4S/HFC-8S only bits:

- Bit 0 0x001 = Use master clock for this S/T interface (ony once per chip).

The master clock is taken from active S/T interface (BRI) automatically. If the multiple ports are TE-mode, the controller automatically selects one as clock source. You may override it by setting this bit on the appropriate port.

- Bit 1 0x002 = transmitter line setup (non capacitive mode)
- Bit 2 0x004 = Disable E-channel. (No E-channel processing)

Don't change these bits. These bits have no practical use.

HFC-E1 only bits:

- Bit 0 0x0001 = interface: 0=copper, 1=optical

Select interface type connected to your HFC-E1 controller. For electrical interface (RJ 45) clear this bit. If there would be a card with optical interface, you may need to set this bit.

- Bit 1 0x0002 = reserved (later for 32 B-channels transparent mode)
- Bit 2 0x0004 = Report LOS
- Bit 3 0x0008 = Report AIS
- Bit 4 0x0010 = Report SLIP
- Bit 5 0x0020 = Report RDI

Use one or more bits to report special interface states and events:

- LOS: Loss Of Signal, will indicate no signal on receive input.
- AIS: Alarm Indication Signal, will indicate continuous 1's for a certain time.
- SLIP: If the jitter buffer overflows, it produces a slip of frame(s).
- RDI: If we receive Remote Defect indication, the remote side has LOS.

These signals are sent via special PH_SIGNAL_IND message to the application.

- Bit 8 0x0100 = Turn off CRC-4 Multiframe Mode, use double frame mode

The most common framing is CRC-4 multi frame. If your remote Interfaces uses double frame mode, set this bit.

- Bit 9 0x0200 = Force get clock from interface, even in NT mode.
- Bit 10 0x0400 = Force put clock to interface, even in TE mode.

By default, TE mode interfaces (when selected by application) will retrieve clock from remote interface. If card is in NT mode, it provides clock to the remote interface. The clock will flow from NT interface to TE interface. I some special cases it may be required to change this behavior. (Example: A leased line, where both ends provide clock from provider, but one side will be connected to NT mode interface.)

- Bit 11 0x0800 = Use direct RX clock for PCM sync rather than PLL.
- Bit 12-13 0xX000 = elastic jitter buffer (1-3), Set both bits to 0 for default.

By default, a jitter attenuation PLL will smooth the jitter. This is required to pass the compliance test. To disable this PLL, set this bit. The PCM will follow jitter and all slave cards too. The elastic jitter buffer is used delay data, so jitter will not cause underrun or overflow. The size is 1 to 3 frames (clock cycles). By default, the buffer is set to two frames.

(all other bits are reserved and shall be 0)

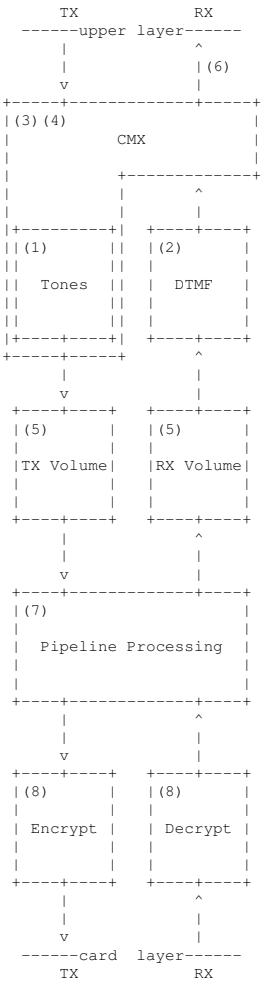
3. mISDN_dsp module:

The hfcmulti driver provides special features for the conference/DTMF/tones module "**mISDN_dsp.o**".

The module realizes all features in software. By using this HFC module, some are supported in hardware. These module is useful in order to use the hardware features, since the HFC module doesn't provide a high level API. Restrictions apply to the following features if they should be realized in hardware. If the hardware cannot handle the features, they will be substituted by software:

- Conferences must be connected on the same chip.
- Up to 8 conferences are possible on one chip. On the HFC-E1 a maximum of 24 parties may be in conferences. There is no limit on the number of parties in one conference.
- Volume control is only possible with some special values.
- Mixing of tones with the conference's sound is not possible. During playing of tones, the party is removed from the conference.

The data flow diagram:



Data is ceiveived from card and decrypted, if crypto is enabled (8). Echo cancellation is performed (7), volume is changed (5) and DTMF digits are decoded (2). In bridfe mode, the data is reclocked (3)(4) and forwarded to other member(s).
Data is transmitted down from CMX (bridge). Tones are added, if enabled (1). The volume is changed (5), the echo cancellation is processed (7). The the data is encrypted, if enabled (8).

4. ioctl():

The hfcmulti driver offer special **ioctl** commands. Use IMCTRLREQ with “struct mISDN_ctrl_req” on B-channel sockets. The structure must be filled with “op”, “p1”, and “p2”.

op = MISDN_CTRL_RX_OFF
p1 = 0 | 1

Will turn receive data off. This will disable FIFO read, if no receive data is required. Set p1 to 0, to enable receive data at any time. This control may be given before PH_ACTIVATE_REQ.

op = MISDN_CTRL_HFC_PCM_CONN
p1 = 0x0BSS (TX)
p2 = 0x0BSS (RX)

Will connect B-channel with PCM, rather than FIFO. The HFC has two banks: 0x000 (bank 1) and 0x100 (bank 2). Also it has 32 slots (0x000 – 0x01f). The number of slots can be increased up to 128, when specified by module options. If 0x0200 is given, the slot is looped for echo generation.

If PCM is connected, receive FIFO is still used to transfer data to user space. Also transmit FIFO will automatically enabled, if data is sent to card. As soon as there is no more data to be sent, the transmit FIFO will be disabled and switched back to PCM.

Example: Two cards use timeslot 5 to interconnect a B-channel. Since we have two banks, we may use one bank for one direction and the other bank for the other direction.

Card 1: op = MISDN_CTRL_HFC_PCM_CONN
Card 1: p1 = 0x0005 (TX)
Card 1: p2 = 0x0105 (RX)

Card 2: op = MISDN_CTRL_HFC_PCM_CONN
Card 2: p1 = 0x0005 (TX)
Card 2: p2 = 0x0105 (RX)

Example: One card use timeslot 8 and 9 to interconnect a B-channel. Since we are on the same card, a B-channel must write to timeslot 8, and the other B-channel must read from timeslot 8. Because only one channel may write to a timeslot, we need another another timeslot for the other direction:

Bchannel A: op = MISDN_CTRL_HFC_PCM_CONN
Bchannel A: p1 = 0x0008
Bchannel A: p2 = 0x0009

Bchannel B: op = MISDN_CTRL_HFC_PCM_CONN
Bchannel B: p1 = 0x0009
Bchannel B: p2 = 0x0008

Example: One card will loop a Bchannel via PCM. One timeslot must be used to write and read. Also this timeslot must be looped, rater than connected to PCM. We use special loop flag (0x0200) and timeslot 3 in this example.

Op = MISDN_CTRL_HFC_PCM_CONN
p1 = 0x0203
p2 = 0x0203

op = MISDN_CTRL_HFC_PCM_DISC

Will disconnect Bchannel from PCM if previously connected.

op = MISDN_CTRL_HFC_CONF_JOIN
p1 = 0…7

Will connect Bchannel to a conference. There are 8 conference engines to select via p1. Also it is required to connect to a looped PCM timeslot. There is no limit of Bchannels in one conferenct. All Bchannels must be on the same card. Each Bchannel requires an individual looped timeslot.

Example: Bchannel A, B, and C are in a conference.

Bchannel A: op = MISDN_CTRL_HFC_PCM_CONN
Bchannel A: p1 = 0x0008
Bchannel A: p2 = 0x0009
Bchannel A: op = MISDN_CTRL_HFC_CONF_JOIN
Bchannel A: p1 = 0…7

op = MISDN_CTRL_HFC_ECHOCAN_ON

Turns hardware echo cancellation on. The card must be equipped with hardware echo canceller and must be supported by this driver.

op = MISDN_CTRL_HFC_ECHOCAN_OFF

Turns hardware echo cancellation off.

5. References:

The module is part of the “mISDN V2” driver. Current source for the mISDN driver is: http://www.mISDN.org.

This hfcmulti and dsp drivers are written by Jolly (Andreas Eversberg) (jolly@eversberg.eu), inspired by the “hfc_pci” driver by Werner Cornelius (info@isdn-development.de) and Karsten Keil, and inspired by the Zaptel channel driver by Kapejod (http://www.junghanns.net). Improvements are made by BeroNet (http://www.beronet.com) and other people.

This module and mISDN_dsp.ko have features developed for the “Linux-Call-Router” project. They are also designed to be useful in other possible projects. The LCR project’s page is http://www.linux-call-router.de.

dsp module Documentation

1. What is what?

The module is used to generate **tones and patterns**, make **cross connections**, make **conferences**, decode **DTMF** and **encrypt** audio data. Everything is possible in software, but may be realized in hardware if it is supported. In contrast to the user space, it has the following benefits:

- Real time processing at kernel space to provide jitter-free audio transfer
- Support of a-Law and µ-Law audio codecs
- DTMF tone generation and decoding
- Pattern and tone generation to indicate call state (dialtone, ringing, busy, …)
- Cross connection of two parties (transparent interconnection)
- Conferences with an unlimited number of parties
- A conference will not affect the audio data from/to the party. Audio data from a party, which is in a conference, can be transferred to the user space without the other parties mixed to it. Audio data to a party, which is in a conference, can be transmitted to the user without the other parties hearing it.
- Volume control of the transmit and receive audio stream (hardware interface side)
- Support of multiple port HFC chips (HFC-4S / HFC-8S / HFC-E1) to provide hardware conference, cross connections, DTMF detection (and sample-loops).
- Blowfish encryption using a special codec to synchronize data blocks.

2. Loading kernel module

Note: The module can be loaded any time. It will provide layer-3 of B-channels for transparent audio.

```
modprobe mISDN_dsp [debug=XXXX] [options=XXX]
```

3. Special PH_CONTROL commands

The PH_CONTROL | REQUEST message will be used to give options to the mISDN_dsp module. The first integer (4 byte) of the message (not the mISDN header) will be used as the subcommand. The bytes must be given in the running CPU byte order. The subcommands are defined in mISDN.h and are explained here:

- DTMF_TONE_START
- DTMF_TONE_STOP

Turn on or off DTMF detection.

- DSP_CONF_JOIN

Join a conference. The next integer (4 bytes) of the message define the conference ID. The ID must be greater 0. All b-channels with the same conference ID will be able to talk to each other.

- DSP_CONF_SPLIT

Slit from a conference.

- DSP_RECEIVE_OFF
- DSP_RECEIVE_ON

Turn off or on receive data from DSP. If supported by driver and if no data is required for DSP itself, it will turn off receive FIFO, to save CPU cycles. If no receive data is required, it should always be turned off.

- DSP_ECHO_ON
- DSP_ECHO_OFF

Turn on or off the local echo. All data received from interface will be echoed back. This also works in conference and during bridging. If supported by hardware, hardware will loop B-channel. This feature is used for delay benchmarking.

- DSP_MIX_ON
- DSP_MIX_OFF

Turn mixing of transmit data and conference data on or off. Mixing is not supported by any chip hardware, so it must be enabled here and will be done in software. When mixing is off, the transmit data causes to “overwrite” the conference data. When mixing is on, the audio waves of transmit data and conference data are added.

- DSP_DELAY
- DSP_JITTER

Please don’t use it, it is not tested and may cause unwanted results.

- DSP_TX_DEJITTER
- DSP_TX_DEJ_OFF

When data is transmitted to DSP, the TX-buffer is filled. To avoid under runs, it is useful to preload the buffer with a certain level and refill it at a rate of 8000 Hz. Sometimes, if data from user space must have low latency (VoIP), the buffer should be as empty as possible, so

delay is low. If DSP_TX_DEJITTER is enabled, the buffer will be used for de-jittering data from user space. If the jitter changes over time, the buffer will automatically raise or lower.

- DSP_TONE_PATT_ON
- DSP_TONE_PATT_OFF

Play or stop pattern/tone. The next 4 bytes of the message define the pattern/tone. See definitions TONE_GERMAN_*, TONE_AMERICAN_* and TONE_SPECIAL_* in mISDN.h. The transmit data is ignored and replaced by the given tone. If tones are turned off, no pattern must be given. To change pattern, the current pattern may not be turned off.

- DSP_VOL_CHANGE_TX
- DSP_VOL_CHANGE_RX

Change the volume of the transmit or receive data. The next 4 bytes of the message define the change of volume. The 4 bytes are signed integer. A value of zero doesn't change the volume. A value of 1 will double, a value of 2 quadruple, and so on. A value of –1 will halve, a value of –2 will quarter, and so on. The range is –8 … 8. Note: A telephone connected via NT-mode receives the TX data and vice versa.

- DSP_BF_ENABLE_KEY

Enable encryption/decryption using blowfish. The next 4-56 byte of the message will specify the key to use. The result will be a message of type PH_CONTROL | INDICATION that has the result flag stored in the first 4 bytes. They are BF_ACCEPT and BF_REJECT, depending on the success of enabling encryption.

- DSP_BF_DISABLE

Disable encryption/decryption.

- DSP_PIPELINE_CFG

<tbd>

4. References:

The new HiSax driver "mISDN" is required to use this module. The module is part of the driver. Current source for the mISDN driver is: http://www.isdn4linux.de.

This driver is written by Jolly (Andreas Eversberg) (jolly@jolly.de).

This module and other modules have features developed for the “PBX4Linux” project. They are also designed to be useful in other possible projects. The PBX project’s page is http://isdn.jolly.de.