NAME

zsh – On-board serial HDLC/SDLC interface

SYNOPSIS

#include <fcntl.h>
open(/dev/zsh*n, mode* **);**
open(/dev/zsh*, mode* **);**

AVAILABILITY

SPARC

DESCRIPTION

The **zsh** module is a loadable STREAMS driver that implements the sending and receiving of data packets as **HDLC** frames over synchronous serial lines. The module is not a standalone driver, but instead depends upon the **zs** module for the hardware support required by all on-board serial devices. When loaded this module acts as an extension to the **zs** driver, providing access to an **HDLC** interface through character-special devices.

The **zsh***n* devices provide what is known as a **data path** which supports the transfer of data via **read**(2) and **write**(2) system calls, as well as **ioctl**(2) calls. Data path opens are exclusive in order to protect against injection or diversion of data by another process.

The **zsh** device provides a separate **control path** for use by programs that need to configure or monitor a connection independent of any exclusive access restrictions imposed by data path opens. Up to three control paths may be active on a particular serial channel at any one time. Control path accesses are restricted to **ioctl**(2) calls only; no data transfer is possible.

When used in synchronous modes, the **Z8530 SCC** supports several options for **clock sourcing** and **data encoding**. Both the transmit and receive clock sources can be set to be the external **T**ransmit **C**lock (**TRxC**), external **R**eceive **C**lock (**RTxC**), the internal **B**aud **R**ate **G**enerator (**BRG**), or the output of the **SCC**'s **D**igital **P**hase-**L**ock **L**oop (**DPLL**).

The **B**aud **R**ate **G**enerator is a programmable divisor that derives a clock frequency from the **PCLK** input signal to the **SCC**. A programmed baud rate is translated into a 16-bit **time constant** that is stored in the **SCC**. When using the **BRG** as a clock source the driver may answer a query of its current speed with a value different from the one specified. This is because baud rates translate into time constants in discrete steps, and reverse translation shows the change. If an exact baud rate is required that cannot be obtained with the **BRG**, an external clock source must be selected.

Use of the **DPLL** option requires the selection of **NRZI** data encoding and the setting of a non-zero value for the baud rate, because the **DPLL** uses the **BRG** as its reference clock source.

A **local loopback mode** is available, primarily for use by the **syncloop**(1M) utility for testing purposes, and should not be confused with **SDLC** loop mode, which is not supported on this interface. Also, an **auto-echo** feature may be selected that causes all incoming data to be routed to the transmit data line, allowing the port to act as the remote end of a digital loop. Neither of these options should be selected casually, or left in use when not needed.

The **zsh** driver keeps running totals of various hardware generated events for each channel. These include numbers of packets and characters sent and received, abort conditions detected by the receiver, receive **CRC** errors, transmit underruns, receive overruns, input errors and output errors, and message block allocation failures. Input errors are logged whenever an incoming message must be discarded, such as when an abort or **CRC** error is detected, a receive overrun occurs, or when no message block is available to store incoming data. Output errors are logged when the data must be discarded due to underruns, **CTS** drops during transmission, **CTS** timeouts, or excessive watchdog timeouts caused by a cable break.

IOCTLS

The **zsh** driver supports several **ioctl( )** commands, including:

S_IOCGETMODE Return a **struct scc_mode** containing parameters currently in use. These include the transmit and receive clock sources, boolean loopback and **NRZI** mode flags and the integer baud rate.

S_IOCSETMODE The argument is a **struct scc_mode** from which the **SCC** channel will be programmed.

S_IOCGETSTATS Return a **struct sl_stats** containing the current totals of hardware-generated events. These include numbers of packets and characters sent and received by the driver, aborts and **CRC** errors detected, transmit underruns, and receive overruns.

S_IOCCLRSTATS Clear the hardware statistics for this channel.

S_IOCGETSPEED Returns the currently set baud rate as an integer. This may not reflect the actual data transfer rate if external clocks are used.

S_IOCGETMCTL Returns the current state of the **CTS** and **DCD** incoming modem interface signals as an integer.

The following structures are used with **zsh ioctl( )** commands:

```
struct scc_mode {
        char    sm_txclock;     /* transmit clock sources */
        char    sm_rxclock;     /* receive clock sources */
        char    sm_iflags;      /* data and clock inversion flags (non-zsh) */
        u_char  sm_config;      /* boolean configuration options */
        int     sm_baudrate;    /* real baud rate */
        int     sm_retval;      /* reason codes for ioctl failures */
};

struct sl_stats {
        long    ipack;          /* input packets */
        long    opack;          /* output packets */
        long    ichar;          /* input bytes */
        long    ochar;          /* output bytes */
        long    abort;          /* abort received */
        long    crc;            /* CRC error */
```

```
        long    cts;            /* CTS timeouts */
        long    dcd;            /* Carrier drops */
        long    overrun;        /* receive overrun */
        long    underrun;       /* transmit underrun */
        long    ierror;         /* input error */
        long    oerror;         /* output error */
        long    nobuffers;      /* receive side memory allocation failure */
};
```

ERRORS

An **open( )** will fail if a **STREAMS** message block cannot be allocated, or:

ENXIO The unit being opened does not exist.

EBUSY The device is in use by another serial protocol.

An **ioctl( )** will fail if:

EINVAL An attempt was made to select an invalid clocking source.

EINVAL The baud rate specified for use with the baud rate generator would translate to a null time constant in the **SCC**'s registers.

FILES

/dev/zsh[0-1],/dev/zsh character-special devices

/usr/include/sys/ser_sync.h header file specifying synchronous serial communication definitions

SEE ALSO

syncinit(1M), **syncloop**(1M), **syncstat**(1M), **ioctl**(2), **open**(2), **read**(2), **write**(2), **zs**(7D)

Refer to the *Zilog Z8530 SCC Serial Communications Controller* for details of the **SCC**'s operation and capabilities.

DIAGNOSTICS

zsh data open failed, no memory, rq=*nnn*

zsh clone open failed, no memory, rq=*nnn*

A kernel memory allocation failed for one of the private data structures. The value of *nnn* is the address of the read queue passed to **open**(2).

zsh_open: can't alloc message block

The open could not proceed because an initial STREAMS message block could not be made available for incoming data.

zsh: clone device *d* **must be attached before use!**

An operation was attempted through a control path before that path had been attached to a particular serial channel.

zsh*n***: invalid operation for clone dev.**

An inappropriate STREAMS message type was passed through a control path. Only **M_IOCTL** and **M_PROTO** message types are permitted.

zsh*n*: not initialized, can't send message

An **M_DATA** message was passed to the driver for a channel that had not been programmed at least once since the driver was loaded. The **SCC**'s registers were in an unknown state. The **S_IOCSETMODE** ioctl command performs the programming operation.

zsh*n*: transmit hung

The transmitter was not successfully restarted after the watchdog timer expired.