NAME

stc – Serial Parallel Communications driver for SBus

DESCRIPTION

The **SPC/S** SBus communications board consists of eight asynchronous serial ports and one *IBM PS/2-compatible* parallel port. The *stc* driver supports up to 8 **SPC/S** boards in an SBus system. Each serial port has full modem control: the **CD**, **DTR**, **DSR**, **RTS** and **CTS** modem control lines are provided, plus flow control is supported in hardware for either **RTS/CTS** hardware flow control or **DC1/DC3** software flow control. The parallel port is unidirectional with support for the **ACK**, **STROBE**, **BUSY**, **PAPER OUT**, **SELECT** and **ERROR** interface signals. Both the serial and parallel ports support those **termio**(7I) device control functions specified by flags in the **c_cflag** word of the **termios**(3) structure; in addition, the serial ports support the **IGNPAR**, **PARMRK**, **INPCK**, **IXON**, **IXANY** and **IXOFF** flags in the **c_iflag** word of the **termios**(3) structure. The latter **c_iflag** functions are performed by the *stc* driver for the serial ports. Since the parallel port is a unidirectional, output-only port, no input **termios**(3) (**c_iflag**) parameters apply to it. Trying to execute a nonsensical **ioctl( )** on the parallel port is not recommended. All other **termios**(3) functions are performed by STREAMS modules pushed atop the driver. When an *stc* device is opened, the **ldterm**(7M) and **ttcompat**(7M) STREAMS modules are automatically pushed on top of the stream if they are specified in the **/etc/iu.ap** file (the default condition), providing the standard **termio**(7I) interface.

The device names of the form **/dev/term/***n* or **/dev/ttyy***n* specify the serial I/O ports provided on the **SPC/S** board, conventionally as incoming lines. The device names of the form **/dev/cua/***n* or **/dev/ttyz***n* specify the serial I/O ports provided on the **SPC/S** board, conventionally as outgoing lines. The device names of the form **/dev/printers/***n* or **/dev/stclp***n* specify the parallel port, and the device name of the form **/dev/stc***n* specify a special control port per board.

To allow a single tty line to be connected to a modem and used for both incoming and outgoing calls, a special feature, controlled by the minor device number, has been added. Minor device numbers in the range *128-191* correspond to the same physical lines as those in the range *0-63* (that is, the same line as the minor device number minus *128*).

A dial-in line has a minor device in the range *0-63* and is conventionally named **/dev/term/***n***,** where *n* is a number indicating which dial-in line it is (so that **/dev/term/0** is the first dial-in line), and the dial-out line corresponding to that dial-in line has a minor device number *128* greater than the minor device number of the dial-in line and is conventionally named **/dev/cua/***n***,** where *n* is the number of the dial-in line. These devices will also have the compatibility names **/dev/ttyz***n***.**

The **/dev/cua/***n* lines are special in that they can be opened even when there is no carrier on the line. Once a **/dev/cua/***n* line is opened, the corresponding **/dev/term/***n* line cannot be opened until the **/dev/cua/***n* line is closed; a blocking open will wait until the **/dev/cua/***n* line is closed (which will drop **DTR**, after which **DCD** will usually drop as well) and carrier is detected again, and a non-blocking open will return an error. Also, if the **/dev/term/***n* line has been opened successfully (usually only when carrier is recognized on the modem) the corresponding **/dev/cua/***n* line can not be opened. This allows a modem to be attached to **/dev/term/0**, for example, and used for dial-in (by enabling the line for login (using **pmadm**(1M)) and also used for dial-out (by **tip**(1) or **uucp**(1C)) as **/dev/cua/0** when no one is logged in on the line.

The parallel port is given the name **/dev/stclp***n***,** where *n* is the **SPC/S** unit number (see **Minor Numbers**, below).

The control port, named **/dev/stc***n***,** where *n* is the *SPC/S*, is available. And **ioctl( )** is provided for this special file which allow the collection of statistics maintained on serial port performance.

Minor Numbers

o p u u | *u l l l* – these correspond to bits in the minor number

- *o* set if this device is an outgoing serial line
- *p* set if this is a parallel port device
- *u* device unit number
- *l* device line number if this is the parallel port line, *'p'* should be 1 and *'lll'* should be all 0's if this is the control line, both *'p'* and *'lll'* should be set to all 1's

IOCTLS

The standard set of **termio ioctl( )** calls are supported by the *stc* driver on both the serial and parallel ports.

If the **CRTSCTS** flag in the **c_cflag** is set and if **CTS** is high, output will be transmitted; if **CTS** is low, output will be frozen. If the **CRTSCTS** flag is clear, the state of **CTS** has no effect. Breaks can be generated by the **TCSBRK**, **TIOCSBRK** and **TIOCCBRK ioctl( )** calls. The modem control lines **TIOCM_CAR**, **TIOCM_CTS**, **TIOCM_RTS**, **TIOCM_DSR** and **TIOCM_DTR** are provided for the serial ports, although the **TIOCMGET ioctl( )** call will not return the state of the **TIOCM_RTS** or **TIOCM_DSR** lines, which are *output-only* signals.

The serial port input and output line speeds may be set to any of the speeds supported by **termio**(7I).

DEVICE-SPECIFIC IOCTLS

The following additional **ioctl( )**'s are supported by the *stc* driver.

STC_SPPC(struct ppc_params_t *)
set parallel port parameters (valid until changed or **close( )**)

STC_GPPC(struct ppc_params_t *)
get parallel port parameters (valid until changed or **close( )**)

```
struct ppc_params_t {
        u_int   flags;          /* driver status flag */
        u_int   state;          /* status of the printer interface */
        u_int   strobe_w;       /* strobe width, in microseconds */
        u_int   data_setup;     /* data setup time, in microseconds */
        u_int   ack_timeout;    /* ACK timeout in secs */
        u_int   error_timeout;  /* PAPER OUT, etc... timeout in secs */
        u_int   busy_timeout;   /* BUSY timeout in seconds */
};
```

The possible values for *flags*, defined in **/usr/include/sys/stcio.h,** are:

PP_PAPER_OUT honor PAPER OUT from port; returned HIGH means PAPER OUT.

PP_ERROR honor ERROR from port; returned HIGH means ERROR.

PP_BUSY honor BUSY from port; returned HIGH means BUSY.

PP_SELECT honor SELECT from port; returned HIGH means OFFLINE.

PP_MSG print console message on every error scan.

PP_SIGNAL send a PP_SIGTYPE (SIGURG) to the process if printer error.

The *state* field contains the current status of the printer interface. It is analogous to the bit order of *flags*, but contains the status the driver maintains, masked by the flags that are set. The result of shifting *state PP_SHIFT* bits to the left is the actual state of the hardware.

The **STC_SPPC** and **STC_GPPC** ioctl calls are understood only by the parallel port.

STC_GSTATS(struct stc_stats_t *)
get or reset driver performance statistics on serial ports

```
struct stc_stats_t {
      u_int     cmd;            /* command */
      u_int     qpunt;          /* punting in stc_drainsilo() */
      u_int     drain_timer;    /* posted a timer in stc_drainsilo() */
      u_int     no_canput;      /* canput() failed in stc_drainsilo() */
      u_int     no_rcv_drain;   /* can't call stc_drainsilo() in stc_rcv() */
      u_int     stc_drain;      /* STC_DRAIN flag set on this line */
      u_int     stc_break;      /* BREAK requested on XMIT via stc_ioctl() */
      u_int     stc_sbreak;     /* start BREAK requested via stc_ioctl() */
      u_int     stc_ebreak;     /* end BREAK requested via stc_ioctl() */
      u_int     set_modem;      /* set modem control lines in stc_ioctl() */
      u_int     get_modem;      /* get modem control lines in stc_ioctl() */
      u_int     ioc_error;      /* bad ioctl() */
      u_int     set_params;     /* call to stc_param() */
      u_int     no_start;       /* can't run in stc_start(); already there */
      u_int     xmit_int;       /* transmit interrupts */
      u_int     rcv_int;        /* receive interrupts */
      u_int     rcvex_int;      /* receive exception interrupts */
      u_int     modem_int;      /* modem change interrupts */
      u_int     xmit_cc;        /* characters transmitted */
      u_int     rcv_cc;         /* characters received */
      u_int     break_cnt;      /* BREAKs received */
      u_int     bufcall;        /* times we couldn't get STREAMS buffer */
      u_int     canwait;        /* stc_drainsilo() called w/pending timer */
      u_int     reserved;       /* this field is meaningless */
};
```

The possible *cmd* values, defined in **/usr/include/sys/stcio.h**, are

STAT_CLEAR clear the line statistics

STAT_GET get the line statistics

The **STC_GSTATS** ioctl works only on the **SPC/S** control port.

SOFTCAR, DTR and CTS/RTS FLOW CONTROL

Several methods may be used to enable or disable *soft carrier* on a particular serial line. The non-programmatic method is to edit the **/kernel/drv/stc.conf** file. For this change to take effect, the machine must be rebooted. See the next section, **SETTING DEFAULT LINE PARAMETERS**, for more information on this method. From within an application program, you can enable or disable the recognition of carrier on a particular line by issuing the **TIOCGSOFTCAR ioctl( )** to the driver.

The default mode of operation for the **DTR** signal is to assert it on the first **open( )** of a serial line and, if *HUPCL* is set, to de-assert it on the last **close( )**. To change the operation of this feature, issue the set on the **/kernel/drv/stc.conf** parameter *flags* field bit **DTR_ASSERT**.

SETTING DEFAULT LINE PARAMETERS

Many default parameters of the serial and parallel ports can be changed using the **/kernel/drv/stc.conf** file. The format of a line in the **stc.conf** file is:

device_tag=*token*[=*value*][:*token*[=*value*]]

For serial ports, the *device_tag* is stc_*n*, where *n* is between 0 and the maximum number of serial ports used by the driver. The token and parameters that follow it apply to both the **/dev/term/***n* entries and **/dev/cua/***n* entries.

For parallel ports, the *device_tag* is stc_p*n*, where *n* is between 0 and the number of parallel ports driven by **stc**.

The *token*[=*value*] specifies a *token*, and if the *token* takes a *value*, the *value* to assigned. Tokens that don't take a value are considered boolean. If boolean tokens don't appear in the **stc.conf** file, they will be cleared by the driver. If these tokens appear in the **stc.conf** file, they will be set by the driver.

Tokens that take parameters must have a parameter specified in the *token=value* couplet in the **stc.conf** file. If no parameter or an invalid parameter is specified, the driver will ignore the token and revert to using the driver's default value.

Tokens for Serial Ports

Valid boolean tokens for serial ports are:

soft_carrier- enables the soft carrier on the specified line. When the soft carrier is set, transitions on the carrier detect line will be ignored.

dtr_assert- causes the **DTR** to be asserted on the next open of the port.

dtr_force- causes **DTR** to be continuously asserted. it overrides any other **DTR** operations and ioctl calls.

dtr_close- use alternate semantics when dealing with **DTR** in close. If this is clear, **DTR** will drop on the close of the port. If this is set, **DTR** will not drop on **close( )** if TS_SOFTCAR (see **termiox**(7I)) is set in the *t_flags*.

cflow_flush- flush any data being held off by remote flow control on **close( )**.

cflow_msg- display a message on the console if data transmission is stalled due to remote flow control blocking the transfer in **close( )**.

instantflow- if transmission is stopped by software flow control and the flow control is disabled via an **ioctl( )** call, the transmitter will be enabled immediately.

display- displays all serial port parameters.

Valid tokens requiring values are:

drain_size- The size of STREAMS buffers allocated when passing data from the receive interrupt handler upstream

hiwater, lowwater- The high water and low water thresholds in the receive interrupt handler 1024 byte buffer

rtpr- The inter-character receive timer

rxfifo- The UART receive fifo threshold.

For the value-carrying tokens for serial ports:

token	default value	min value
hiwater	1010 bytes	2 bytes
lowwater	512 bytes	2 bytes
drain_size	64 bytes	4 bytes
rtpr	18 millisecs	1 millisecs
rxfifo	4 bytes	1 bytes

Tokens for Parallel Ports

Valid boolean tokens for parallel ports are

paper_out- If set, the PAPER OUT signal from the port is monitored. If clear, the signal is ignored.

error- Monitor the ERROR signal from the port. Ignore the signal if clear.

busy- Monitor the BUSY signal from the port. Ignore the signal if clear.

select- Monitor the SELECT, or ON LINE, signal from the port. Ignore the signal if clear.

pp_message- If this token is clear, a console message will be printed when any of the above four enabled conditions are detected, and another when the condition is cleared. If set, a console message will be printed every 60 seconds until the condition is cleared.

pp_signal- If this token is set, the parallel port's controlling process will get a PP_SIGTYPE signal whenever one of the above four conditions is detected. PP_SIGTYPE is defined in **stcio.h**, which is available to the user.

Valid tokens requiring parameters for the parallel ports are

ack_timeout- The amount of time in seconds to wait for an ACK from the port after asserting STROBE and transferring a byte of data.

error_timeout- Amount of time in seconds to wait for an error to go away.

busy_timeout- The amount of time in seconds to wait for a BUSY signal to clear, or zero for an infinite BUSY timeout.

data_setup- The amount of time in microseconds between placing data ont the parallel lines and asserting the STROBE.

strobe_width- width of the STROBE pulse, in microseconds.

For value-carrying tokens for parallel ports:

token	**default value**	**min value**
strobe_width	2 microsecs	1 microsecs
data_setup	2 microsecs	0 microsecs
ack_timeout	60 seconds	5 seconds
errror_timeout	5 seconds	1 seconds
busy_timeout	10 seconds	0 seconds

PARALLEL PORT PARAMETERS

The default values of certain parallel port parameters that govern data transfer between the **SPC/S** board and the device attached to the parallel port will usually work well with most devices; however, some devices don't strictly adhere to the *IBM PS/2-compatible (Centronics-compatible)* data transfer and device control/status protocol, and may require modification of one or more of the default parallel port parameters. Some printers, for example, have non-standard timing on their **SELECT** line, which manifests itself if you start sending data to the printer and then take it off line; when you put it back on line, the printer will not assert it's **SELECT** line until after the next character is sent to the printer. Since the *stc* driver will not send data to the device if it's **SELECT** line is de-asserted, a deadlock condition occurs. To remedy this situation, you can change the default signal list that the *stc* driver monitors on the parallel port by removing the **SELECT** signal from the list. This can be done either through the **/kernel/drv/stc.conf** configuration file or programmatically through the **STC_SPPC ioctl( )** call.

LOADABLE ISSUES

If you try to unload the driver, and one or more of the ports on one or more of the **SPC/S** boards is in use (i.e. **open( )**) by a process, the driver will not be unloaded, and all lines on all **SPC/S** boards, **with the exception of the control ports,** will be marked with an *open inhibit* flag to prevent further opens until the driver is sucessfully unloaded.

ERRORS

An **open( )** will fail with **errno** set to:

ENXIO The unit being opened does not exist.

EBUSY The dial-out device is being opened and the dial-in device is already open, the dial-in device is being opened with a no-delay open and the dial-out device is already open or the unit has been marked as exclusive-use by another process with a **TIOCEXCL ioctl( )** call.

EINTR The open was interrupted by the delivery of a signal.

EPERM The control port for the board was opened by a process whose *uid* was not **root**.

An **ioctl( )** will fail with **errno** set to:

ENOSR A STREAMS data block couldn't be allocated to return data to the caller.

EINVAL An invalid value was passed as the data argument to the **ioctl( )** call or an invalid argument or *op-field* was passed in one of the driver-specific **ioctl( )**'s.

EPERM An **STC_GSTATS ioctl( )** was requested by a process whose *uid* was not **root**.

ENOTTY An unrecognized **ioctl( )** command was received.

FILES

/dev/term/[00-3f]	
/dev/ttyy[00-3f]	hardwired and dial-in tty lines
/dev/cua/[00-3f]	
/dev/ttyz[00-3f]	dial-out tty lines
/dev/printers/[0-7]	
/dev/stclp[0-7]	parallel port lines
/dev/stc[0-7]	control port
/usr/include/sys/stcio.h	header file with **ioctl( )**'s supported by this driver

SEE ALSO

tip(1), **uucp**(1C), **pmadm**(1M), **termios**(3), **mcpp**(7D), **termio**(7I), **termiox**(7I), **ldterm**(7M), **ttcompat**(7M), **allocb**(9F), **bufcall**(9F)

DIAGNOSTICS

All diagnostic messages from the driver appear on the system console. There are three severity levels of messages displayed:

FATAL - the device driver does not get loaded and any **SPC/S** boards installed in the system are inaccessible (usually occurs during the process of **modload**'ing the driver).

ERROR - some condition has caused the normal operation of the board and/or device driver to be disrupted; data loss may or may not occur; this class of message might indicate an impending hardware failure.

ADVISORY -
the device driver has detected a condition that may be of interest to the user of the system; usually this is a transient condition that clears itself.

Messages During Initialization Of Driver/Board:

stc_attach: can't allocate memory for unit structs
FATAL. **kmem_zalloc( )** failed to allocate memory for the driver's internal data structures.

stc_attach: board revision undeterminable
FATAL. The driver did not get a hardware revision level from the board's onboard FCode PROM.

stc_attach: board revision 0x%x not supported by driver.
FATAL. This revision of the board is not supported by the driver.

stc_attach: oscillator revision undeterminable
FATAL. The driver did not get an oscillator revision level from the board's onboard FCode PROM.

stc_attach: wierd oscillator revision (0x%x), assuming 10Mhz
ADVISORY. The board's onboard FCode PROM returned an unanticipated baud-rate oscillator value, so the driver assumes that a 10Mhz oscillator is installed.

stc_attach: error initializing stc%d
FATAL. An error occured while trying to initialize the board; perhaps a memory access failed.

stc_attach: bad number of interrupts: %d
FATAL. An incorrect number of interrupts was read from the board's onboard FCode PROM.

stc_attach: bad number of register sets: %d
FATAL. An incorrect number of register sets was read from the board's onboard FCode PROM.

stc_init: stc%d GIVR was not 0x0ff, was: 0x%x
FATAL. The *cd-180* 8-channel UART failed to initialize properly, or a memory fault occured while trying to access the chip.

cd180_init: stc%d GIVR was not 0x0ff, was: 0x%x
FATAL. The *cd-180* 8-channel UART failed to initialize properly, or a memory fault occured while trying to access the chip.

stc%d: board revision: 0x%x should be updated
ADVISORY. Two versions of the FCode PROM on the **SPC/S** card have been released; V1.0 (0x4) and V1.1 (0x5). The V1.1 PROM fixes some incompatibilities between the V1.0 FCode PROM (on the **SPC/S**) and the V2.0 *OpenBOOT* PROM (on your system) and is required on an **SPC/S** card to be used in a system running Solaris 2.X.

stc%d: system boot PROM revision V%d.%d should be updated
ADVISORY. Your system's BOOT PROM should be updated to at least V1.3 because prior versions of the BOOT PROM did not map the SBus interrupt levels that the **SPC/S** uses correctly.

Messages Related To The Serial Port:

SET_CCR: CCR timeout
ERROR. the *cd-180*'s **CCR** register did not return to zero within the specified timeout period after it was issued a command

PUTSILO: unit %d line %d soft silo overflow
ERROR. The driver's internal receive data silo for the enunciated line has overflowed because the system has not gotten around to pulling data out of the silo; check that you are using the correct flow control; all data in the silo is

flushed; *this message may also frequently appear due to a hardware crosstalk problem that was fixed in later releases of the board*.

stc_rcvex: unit %d line %d receiver overrun, char: 0x%x
ERROR. The driver could not get around to service the *cd-180* receive data interrupt before the *cd-180*'s receive data fifo filled up; *this message may also frequently appear due to a hardware crosstalk problem that was fixed in later releases of the board*.

stc_drainsilo: unit %d line %d can't allocate streams buffer
ERROR. The driver could not get a STREAMS message buffer from **bufcall**(9F); all data in the driver's receive data silo is flushed.

stc_drainsilo: unit %d line %d punting put retries
ERROR. After trying several times to send data down the stream from the driver to the application and finding the path blocked, the driver gives up; all data in the driver's receive data silo is flushed.

stc_modem: unit %d line %d interesting modem control
ADVISORY. The *cd-180* posted a modem control line change interrupt, but upon examination by the driver, no modem control lines had changed state since the last time a scan was conducted; if you see this problem frequently, it is likely that your data cables are either too long or picking up induced noise.

Messages Related To The Parallel Port:

ppc_stat: unit %d PAPER OUT
ADVISORY. The device connected to the parallel port on the enumerated BOARD has signalled that it's out of paper (**PAPER OUT** line asserted).

ppc_stat: unit %d PAPER OUT condition cleared
ADVISORY. The previously-detected paper out condition has been cleared by the device connected to the parallel port on the enumerated board (**PAPER OUT** line de-asserted)

ppc_stat: unit %d OFFLINE
ADVISORY. The device connected to the parallel port on the enumerated board has signalled that it is off-line (**SLCT** line de-asserted).

ppc_stat: unit %d OFFLINE condition cleared
ADVISORY. The previously-detected off line condition has been cleared by the device connected to the parallel port on the enumerated board (**SLCT** line asserted).

ppc_stat: unit %d ERROR
ADVISORY. The device connected to the parallel port on the enumerated board has signalled that it has encountered an error of some sort (**ERROR** line asserted).

ppc_stat: unit %d ERROR condition cleared
ADVISORY. The previously-detected error condition has been cleared by the device connected to the parallel port on the enumerated board (**ERROR** line de-asserted).

ppc_acktimeout: unit %d ACK timeout
ERROR. The **ACK** line from the device connected to the parallel port did not assert itself within the configurable timeout period; check to be sure that the device is connected and powered on.

ppc_acktimeout: unit %d BUSY timeout
ERROR. The **BUSY** line from the device connected to the parallel port did not de-assert itself within the configurable timeout period; check to be sure that the device is connected and powered on.

ppc_int: unit %d stray interrupt
ADVISORY. The parallel port controller (**ppc**) chip generated an interrupt while the device was closed; this was unexpected and if you see it frequently, your parallel cable might be picking up induced noise causing the **ppc** to generate an unwanted interrupt, or this could indicate that the **ppc** might have an internal problem.

ppc_acktimeout: unit %d can't get pointer to read q
ERROR. Somehow the driver's internal **ppc** data structure became corrupted; this should not happen.

ppc_acktimeout: unit %d can't send M_ERROR message
ERROR. The driver can't send an **M_ERROR** STREAMS message to the application; this should not happen either.

ppc_signal: unit %d can't get pointer to read q
ERROR. Somehow the driver's internal **ppc** data structure became corrupted; this should also not happen.

ppc_signal: unit %d can't send M_PCSIG(PP_SIGTYPE 0x%x) message
ERROR. The driver can't send an **M_PCSIG** STREAMS message to the application (which could cause a signal to be posted); this should also not happen either.

Messages Related To STREAMS Processing:

stc_wput: unit %d trying to M_STARTI on ppc or control device
ADVISORY. An **M_STARTI** STREAMS message was sent to the parallel port or the board control device; this should only happen if an application explictly sends this message.

stc_wput: unit %d line %d unknown message: 0x%x
ADVISORY. An unknown STREAMS message was sent to the driver; check your application coding.

stc_start: unit %d line %d unknown message: 0x%x
ADVISORY. An unknown STREAMS message was sent to the driver; check your application coding.

Messages Related To Serial Port Control:

stc_ioctl: unit %d line %d can't allocate streams buffer for ioctl
ERROR. The driver could not get a STREAMS message buffer from **bufcall( )** for the requested ioctl; the ioctl will not be executed

stc_ioctl: unit %d line %d can't allocate STC_DCONTROL block
ERROR. The driver could not allocate a data block from **allocb**(9F) for the **STC_DCONTROL** return value; the ioctl does not get executed.

stc_ioctl: unit %d line %d can't allocate STC_GPPC block
ERROR. The driver could not allocate a data block from **allocb( )** for the **STC_GPPC** return value; the ioctl does not get executed

stc_ioctl: unit %d line %d can't allocate TIOCMGET block
ERROR. The driver could not allocate a data block from **allocb( )** for the **TIOCMGET** return value; the ioctl does not get executed

stc_vdcmd: unit %d cd-180 firmware revision: 0x%x
ADVISORY. The firmware revision level of the *cd-180*, displayed when the driver is first loaded.