struct termios

Data Structure

data structure containing terminal information

SYNOPSIS

SYNOPSIS

```
#include <termios.h>

struct termios {
        tcflag_t c_iflag;
        tcflag_t c_oflag;
        tcflag_t c_cflag;
        tcflag_t c_lflag;
        cc_t c_cc[NCCS];
        speed_t c_ispeed;
        speed_t c_ospeed;
};
```

DESCRIPTION

The `termios` general terminal interface provides an interface to asynchronous communications devices. The N*u*TCRACKER Platform supports this interface for serial communication ports. This interface is also supported on the console window although the hardware-specific parts obviously do not apply.

Canonical Mode Input Processing

In canonical mode input processing input is processed in units of lines. A line is delimited by a '\n' character or and end-of-file (`EOF`) character. A read request does not return until an entire line is read from the port or a signal is received. Also no matter how many bytes have been requested in the read call at most one line is returned. It is not necessary, however to read a whole line at once; any number of bytes even one may be requested without losing information.

If `MAX_CANON` is defined for the device it is a limit on the number of bytes in a line. The behavior of the system when this limit is exceeded is implementation-dependent. If `MAX_CANON` is not defined there is no such limit. Use the `fpathconf()` function to get the `MAX_CANON` setting for a device.

Erase and kill processing occurs when either of two special characters, the `ERASE` and `KILL` characters is received. This processing affects data in the canonical input queue that has not yet been delimited by a `'\n'` or `EOF` character. This un-delimited data makes up the current line. The `ERASE` character deletes the last character in the current line if any. The `KILL` character deletes all data in the current line. The `ERASE` and `KILL` characters have no effect if there is no data in the line and are themselves never placed in the input queue.

Non-canonical Mode Input Processing

In non-canonical mode input processing input data is not assembled into lines and `ERASE` and `KILL` processing do not occur. The values of the `MIN` and `TIME` members of the `c_cc` array of the `termios` structure are used to determine how to process the bytes received.

`MIN` represents the minimum number of bytes that should be received when the `read()` function returns successfully. `TIME` is a timer of 0.1 second granularity (or as close to that value as can be accommodated) that is used to time out bursty and short-term data transmissions. If `MIN` is greater than `MAX_INPUT` the response to the request is undefined. The four possible values for `MIN` and `TIME` and their interactions are as follows:

Case A: `MIN>0 TIME>0`

In this case `TIME` serves as an inter-byte timer and is activated after the first byte is received. As soon as the first byte is received the inter-byte timer is started. As long as bytes keep coming before the timer expires and the total number of bytes does not exceed the amount requested by the read call the read blocks. Since `TIME` is an inter-byte timer it is reset upon receipt of each byte. If the timer expires or the number of bytes requested is read the read returns. It is possible to block indefinitely here since the timer is not started until the first byte is received.

Case B: `MIN>0 TIME=0`

In this case there is no timer. A pending read is not satisfied until `MIN` bytes are received or a signal occurs. If some data has been read before the signal it is returned. If not -1 is returned with `errno` set to `EINTR`.

Case C: `MIN=0 TIME>0`

In this case TIME is a total read timeout. The read returns if either MIN bytes are received before the timer expires or if the timer expires. In the latter case the number of bytes received is returned even if that number is zero. If an interrupt occurs before the timer expires and no bytes are read the read returns -1 with errno set to EINTR.

Case D: MIN=0 TIME=0
: This implements a poll read. The minimum of the number of bytes requested and the number of bytes currently available is returned, without waiting for more bytes to be input. If no characters are available zero is returned. This is not exactly equivalent to the non-blocking case. In the latter the read waits for a brief time for data to be input if none is currently available and sets errno to EAGAIN and returns -1 if no data is available.

Special Characters

Certain characters have special functions on input or output or both. These functions are summarized as follows:

INTR
: Special character on input which is recognized if the ISIG flag is set. Generates a SIGINT signal which is sent to all processes in the process group for which the terminal is the controlling terminal. If ISIG is set the INTR character is discarded when processed.

QUIT
: Special character on input which is recognized if the ISIG flag is set. Generates a SIGQUIT signal which is sent to all processes in the process group for which the terminal is the controlling terminal. If ISIG is set the QUIT character is discarded when processed.

ERASE
: Special character on input which is recognized if the ICANON flag is set. It erases the last character in the current line. It does not erase beyond the start of a line as delimited by an NL, EOF or EOL character. If ICANON is set, the ERASE character is discarded when processed.

KILL
: Special character on input which is recognized if the ICANON flag is set. It deletes the entire line as delimited by the NL, EOF or EOL character. If ICANON is set, the KILL character is discarded when processed.

EOF
: Special character on input which is recognized if the ICANON flag is set. When received all of the bytes waiting to be read are immediately passed to the process without waiting for a newline and the EOF is discarded. Thus is there are no bytes waiting (EOF occurred at the beginning of a line) a byte count of zero shall be returned from read(), representing an end-of-file indication. If ICANON is set, the EOF character is discarded when processed.

NL
: Special character on input, which is recognized if the ICANON flag is set. It is the line delimiter ('\n'). It cannot be changed.

EOL
: Special character on input, which is recognized if the ICANON flag is set. It is an additional line delimiter like NL.

STOP
: Special character on both input and output which is recognized if the IXON (output control) or IXOFF (input control) flag is set. It can be used to temporarily suspend output. If IXON is set the STOP character is discarded when processed.

START
: Special character on both input and output which is recognized if the IXON (output control) or IXOFF (input control) flag is set. Can be used to resume output that has been suspended by a STOP character. If IXON is set the START character is discarded when processed.

CR
: Special character on input, which is recognized if the ICANON flag is set; it is the '\r' as denoted in the C Standard. When ICANON and ICRNL are set and IGNCR is not set this character is translated into a NL and has the same effect as a NL character. This character cannot be changed.

The NL and CR characters cannot be changed. It is device-specific if the START and STOP characters can be changed. The values for INTR QUIT, ERASE KILL and EOF are changeable to suit user preferences.

If two or more special characters have the same value the results are unspecified.

The special control characters values are defined by the array `c_cc` in the `termios` structure. The subscript name and description for each element in both canonical and non-canonical modes are shown in the following table.

Subscript Usage Canonical Mode	**Non-Canonical Mode**	**Description**
VEOF		EOF character
VEOL		EOL character
VERASE		ERASE character
VINTR	VINTR	INTR character
VKILL		KILL character
	VMIN	MIN value
VQUIT	VQUIT	QUIT character
	VTIME	TIME value
VSTART	VSTART	START character
VSTOP	VSTOP	STOP character

The subscript values are unique.

If the value of one of the changeable special control characters is `_POSIX_VDISABLE` that function shall be disabled, that is, no input data is recognized as the disabled special character. If `ICANON` is not set the value of `_POSIX_VDISABLE` has no special meaning for the `VMIN` and `VTIME` entries of the `c_cc` array.

Input Modes

Values of the `c_iflag` field describe the basic terminal input control and are composed of the bitwise inclusive-OR of the masks shown.

Mask Name	**Description**
BRKINT	Signal interrupt on break
ICRNL	Map CR to NL on input
IGNBRK	Ignore break condition
IGNCR	Ignore CR
IGNPAR	Ignore characters with parity errors
INLCR	Map NL to CR on input
INPCK	Enable input parity check
ISTRIP	Strip character
IXOFF	Enable start/stop input control
IXON	Enable start/stop output control
PARMRK	Mark parity errors

In the context of asynchronous serial data transmission a break condition is defined as a sequence of zero-valued bits that continues for more than the time to send one byte. The entire sequence of zero-valued bits is interpreted as a single break condition even if it continues for a time equivalent to more than one byte.

If IGNBRK is set a break condition detected on input is ignored that is not put on the input queue and therefore not read by any process. If IGNBRK is not set and BRKINT is set the break condition shall flush the input and output queues and if the terminal is the controlling terminal of a process group the break condition generates a single SIGINT signal to that process group. If IGNBRK is not set it is illegal for BRKINT not to be set; this is silently enforced by the system. POSIX.1 specifies that this condition shall cause the break to be read as a single null character or a combination of characters depending on the PARMRK setting; this is not supported by the N*u*TCRACKER Platform.

If IGNPAR is set a byte with a framing or parity error (other than break) is ignored.

PARMRK and ISTRIP are not supported. The system silently enforces settings of zero for these two flags. POSIX.1 specifies that ISTRIP causes valid input bytes to be first stripped to seven bits.

If INPCK is set input parity checking is enabled. If INPCK is not set input parity checking is disabled allowing output parity generation without input parity errors. Note that whether input parity checking is enabled or disabled is independent of whether parity detection is enabled or disabled. If parity detection is enabled but input parity checking is disabled the hardware to which the terminal is connected shall recognize the parity bit but the terminal driver shall not check whether this bit is set correctly or not.

If INLCR is set a received NL character is translated to a CR character. If IGNCR is set a received CR character is ignored (not read). If IGNCR is not set and ICRNL is set a received CR character is translated into a NL character.

If IXON is set start/stop output control is enabled. A received STOP character shall suspend output and a received START character shall restart output. When IXON is set START and STOP characters are not read but merely perform flow-control functions. When IXON is not set the START and STOP characters are read.

If IXOFF is set *start*/*stop* input control is enabled. The system shall transmit one or more STOP characters which are intended to cause the terminal device to stop transmitting data as needed to prevent the input queue from overflowing and causing loss of data and shall transmit one or more START characters which shall cause the terminal device to resume transmitting data, as soon as the device can continue transmitting data without risk of overflowing the input queue.

Output Modes

Values of the c_oflag field describe the basic terminal output control and are composed of the bitwise inclusive-OR of the following masks which shall be bitwise distinct:

Mask Name	Description
OPOST	Perform output processing
OLCUC	Map lower case to upper on output
ONLCR	Map NL to CR-NL on output
OCRNL	Map CR to NL on output
ONOCR	No CR output at column 0
ONLRET	NL performs CR function
OFILL	Use fill characters for delay
OFDEL	Fill is DEL else NUL.
NLDLY	Select new-line delays: NL0 NL1
CRDLY	Select carriage-return delays: CR0 CR1 CR2 CR3

TABDLY	Select horizontal-tab delays: TAB0 TAB1 TAB2
XTABS	Expand tabs to spaces
BSDLY	Select backspace delays: BS0 BS1
VTDLY	Select vertical tab delays: VT0 VT1
FFDLY	Select form-feed delays: FF0 FF1

If OPOST is set output characters are post-processed as indicated by the remaining flags otherwise characters are transmitted without change.

If OLCUC is set a lower-case alphabetic character is transmitted as the corresponding upper-case character. This function is often used in conjunction with IUCLC in standard UNIX versions. (IUCLC is not supported by the N*u*TCRACKER Platform).

If ONLCR is set the NL character is transmitted as the CR-NL character pair. If OCRNL is set the CR character is transmitted as the character. If ONOCR is set no CR character is transmitted when at column 0 (first position). If ONLRET is set the NL character is assumed to do the carriage-return function; the column pointer is set to 0 and thed delays specified for CR are used. Otherwise the NL character is assumed to do just the line-feed function; the column pointer remains unchanged. The column pointer is also set to 0 if the CR character is actually transmitted.

The N*u*TCRACKER Platform does not support the concept of column positions. ONOCR is thus not supported. The current implementation does not support delays (discussed below). Thus ONLRET is also not supported.

The delay bits specify how long transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. In all cases a value of 0 indicates no delay. If OFILL is set fill characters are transmitted for delay instead of a timed delay. This is useful for high baud rate terminals that need only a minimal delay. If OFDEL is set the fill character is DEL, otherwise NUL.

If a form-feed or vertical-tab delay is specified it lasts for about 2 seconds.

New-line delay lasts about 0.10 seconds. If ONLRET is set the RETURN delays are used instead of the NEWLINE delays.

If OFILL is set two fill characters are transmitted.

Carriage-return delay type 1 is dependent on the current column position, type 2 is about 0.10 seconds and type 3 is about 0.15 seconds. If OFILL is set delay type 1 transmits two fill characters and type 2 four fill characters.

Horizontal-tab delay type 1 is dependent on the current column position. Type 2 is about 0.10 seconds. Type 3 specified by TAB3 or XTABS specifies that TAB characters are to be expanded into SPACE characters. If OFILL is set, two fill characters are transmitted for any delay.

Backspace delay lasts about 0.05 seconds. If OFILL is set one fill character is transmitted.

The actual delays depend on line speed and system load.

As mentioned earlier delays are not supported in the current implementation.

Control Modes

Values of the c_cflag field displayed in the following table describe the basic terminal hardware control and are composed of the bitwise inclusive-OR of the masks shown.

Mask Name	**Description**
CLOCAL	Ignore modem status lines
CREAD	Enable receiver
CSIZE	Number of bits per byte CS5 CS6 CS7 CS8

CSTOPB	Send two stop bits else one
HUPCL	Hang up on last close
PARENB	Parity enable
PARODD	Odd parity else even

The CSIZE specify the byte size in bits for both transmission and reception. This size does not include the parity bit if any. Supported byte sizes are 5 6 7 and 8.

If CSTOPB is set two stop bits are used; otherwise one stop bit is used.

If CREAD is set the receiver is enabled; otherwise no characters shall be received. This is always enabled by the N*u*TCRACKER Platform.

If PARENB is set parity generation and detection is enabled and a parity bit is added to each character. If parity is enabled PARODD specifies odd parity if set; otherwise, even parity is used.

If HUPCL is set the modem control lines for the port shall be lowered when the last process with the port open closes the port or the process terminates. The modem connection shall be broken. HUPCL is not supported by the N*u*TCRACKER Platform and silently forced to zero.

If CLOCAL is set a connection does not depend on the state of the modem status lines. If CLOCAL is clear, the modem status lines shall be monitored. The CLOCAL flag has meaning only if the terminal device file is specified as modem-controlled (for example, /dev/com/*n*M) For non-modem controlled devices CLOCAL setting has no effect.

Local Modes

Values of the c_lflag field shown in the following table describe the control of various functions and are composed of the bitwise inclusive-OR of the masks shown.

Mask Name	**Description**
ECHO	Enable echo
ECHOE	Echo ERASE as an error-correcting backspace
ECHOK	Echo KILL
ECHONL	Echo \n
ICANON	Canonical input (erase and kill processing)
IEXTEN	Enable extended functions
ISIG	Enable signals
NOFLSH	Disable flush after interrupt, quit or suspend
TOSTOP	Send SIGTTOU for background output

If ECHO is set input characters are echoed back to the terminal. If ECHO is not set, input characters are not echoed.

If ECHOE and ICANON are set the ERASE character shall cause the terminal to erase the last character in the current line from the display if possible. If there is no character to erase the implementation does nothing.

If ECHOK and ICANON are set, the KILL character shall either cause the terminal to erase the line from the display or shall echo the '\n' character after the KILL character.

If ECHONL and ICANON are set the '\n' character shall be echoed even if the ECHO is not set.

If ICANON is set canonical processing is enabled. This enables the erase and kill edit functions and the assembly of input characters into lines delimited by NL and EOF.

If ICANON is not set read requests are satisfied directly from the input queue. A read shall not be satisfied until at least MIN bytes have been received or the timeout value TIME has expired between bytes. The time value represents tenths of seconds.

If ISIG is set each input character is checked against the special control character INTR and QUIT. If an input character matches one of these control character the function associated with that character is performed. If ISIG is not set, no checking is done. Thus these special functions are possible only if ISIG is set.

IEXTEN NOFLSH and TOSTOP are not implemented.

Input and Output Baud Rates

The baud rates for a com port should be set using one of these functions: cfgetispeed() cfgetospeed(), cfsetispeed() cfsetospeed()>, or by setting the c_ispeed and c_ospeed fields of the termios structure. The N*u*TCRACKER Platform does not support setting baud rates in the c_cflag field.

The input and output baud rates are stored in the termios structure. The values shown in the following table are supported.

Name	Description	Name	Description
B50	50 baud	B75	75 baud
B110	110 baud	B134	134 baud
B150	150 baud	B200	200 baud
B300	300 baud	B600	600 baud
B1200	1200 baud	B1800	1800 baud
B2400	2400 baud	B4800	4800 baud
B7200	7200 baud	B9600	9600 baud
B14400	14400 baud	B19200	19200 baud
B38400	38400 baud	B56K	56000 baud
B57600	57600 baud	B115200	115200 baud
B128K	128000 baud	B256000	256000 baud

Windows does not support different baud rates for input and output. If the input speed is not zero its value is used to set the underlying device baud rate. If not the value of output speed is used. Also the fact that the above values mentioned in the table are available does not mean that all of them can be used in any situation or environment. This also depends on certain other device settings apart from the type of hardware.

Supported ioctls

TCGETS
: The argument is a pointer to a termios structure. The current terminal parameters are fetches and stored in that structure.

TCSETS
: The argument is a pointer to a termios structure. The current terminal parameters are set from the values stored in that structure. The change is immediate.

TCSETSW
: The argument is a pointer to a termios structure. The current terminal parameters are set from the values stored in that structure. The change occurs after all characters queued for output have been transmitted. This form should be used when changing parameters that affect output.

`TCSETSF`
: The argument is a pointer to a `termios` structure. The current terminal parameters are set from the values stored in that structure. The change occurs after all characters queued for output have been transmitted; all characters queued for input are discarded and then the change occurs.

`TCSBRK`
: The argument is an `int` value. Wait for the output to drain. If the value is 0, then send a break (zero-valued bits for 0.25 seconds).

`TCXONC`
: The argument is an `int` value. If the argument is `TCOOFF` suspend output; if `TCOON` restart suspended output; if `TCIOFF` suspend input; if `TCION` restart suspended input.

`TCFLSH`
: The argument is an `int` value. If the argument is `TCIFLUSH` flush the input queue; if `TCOFLUSH`, flush the output queue; if `TCIOFLUSH` flush both the input and output queues.

CONFORMANCE

UNIX 98 with exceptions.

PORTING ISSUES

The N*u*TCRACKER Platform only implements a subset of the UNIX 98 `termios` settings as described in the *DESCRIPTION* section.

AVAILABILITY

PTC MKS Toolkit for Professional Developers
PTC MKS Toolkit for Professional Developers 64-Bit Edition
PTC MKS Toolkit for Enterprise Developers
PTC MKS Toolkit for Enterprise Developers 64-Bit Edition

SEE ALSO

Functions:
`cfgetispeed()`, **`cfgetospeed()`**, **`cfmakeraw()`**, **`cfsetispeed()`**, **`cfsetospeed()`**, **`ioctl()`**, **`tcdrain()`**, **`tcflow()`**, **`tcflush()`**, **`tcgetattr()`**, **`tcgetpgrp()`**, **`tcsendbreak()`**, **`tcsetattr()`**, **`tcsetpgrp()`**

Miscellaneous:
`struct termiox`