How to use TTY with User Terminal

How to use TTY with User Terminal

Stable: 17.03.2020 - 14:36 / Revision: 25.02.2020 - 15:13

Contents

1 Purpose

This article describes how to use TTY with a user terminal. The TTY overview is described in Serial TTY overview article.

The use case of the following examples is a data transfer between a STM32 MPU board and PC, over a USB to a RS232 adapter cable.
The setup of this use case is described in details in the How to get Terminal article.

For the following examples:

- uart4 is activated by default (for the Linux console)
- usart3 is enabled by device tree
- The usart3 pins are connected to a RS232 card
- The RS232 card is connected to the PC over the USB to RS232 adapter cable.

Note: Some TTY tools are used in this article. A list of TTY tools is defined a dedicated article [TTY Tools].

2 Print the file name of the terminal connected to standard input (with tty tool)

```
Board $> tty
# The console is connected to uart4 (aka ttySTM0) #
/dev/ttySTM0
```

3 Change serial port configuration (with stty tool)

Many serial port properties can be displayed and changed with the stty tool. The full feature list is available in stty user manual pages[1].

```
Board $> stty --help
```

- Display the current configuration:

The termios default configuration is specific to each Linux distribution. Before starting a serial communication between two devices, it is recommended to check that termios configurations are compatible on both devices. The termios configurations need to be aligned first.

```
Board $> stty -a -F /dev/ttySTM1
# Display the configuration of uart3 (aka ttySTM1) #
speed 115200 baud; rows 45; columns 169; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>; eol2 = <undef>;
swtch = <undef>; start = ^Q;
stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W; lnext = ^V;
discard = ^O; min = 1; time = 0;
-parenb -parodd -cmspar cs8 hupcl -cstopb cread clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff -iuclc -
ixany -imaxbel -iutf8
opost -olcuc ocrnl -onlcr -onocr onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig -icanon iexten -echo echoe echok -echonl -noflsh -xcase -tostop -echoprt echoctl
echoke -flusho -extproc
```

- Display only the current baud rate:

```
Board $> stty -F /dev/ttySTM1 speed
# uart3 (aka ttySTM1) baud rate is set to 115200 bps #
115200
```

- Change the baud rate:

stty -F /dev/ttySTMx *EXPECTED_BAUDRATE*

Example: change the baud rate to 19200

```
# Change uart3 (aka ttySTM1) baud rate to 19200 bps #
Board $> stty -F /dev/ttySTM1 19200
```

The stty tool proposes many arguments allowing many operations on a tty terminal, such as:

- special settings (various arguments such as speed, line discipline, minimum number of characters for a completed read, size, timeout, etc...)

- control settings
- input settings
- output settings
- local settings
- combination settings

Note: If you want to go further, an interesting tutorial describes termios and stty [2].

4 Send / Receive data (with stty, minicom, echo and cat tools)

Serial counters can be very useful to debug the following use cases.

4.1 Default configuration (8 data bits frame, no parity errors detection, no framing errors detection)

Canonical mode, input modes and output modes termios settings have a major influence on data processing. The following settings can be deactivated for testing.
In case of unexpected behavior, all canonical mode, input modes and output modes settings must be checked. mkssoftware proposes an enriched version of termios manual [3], where the following definitions are provided.

- echo: Enable echo. If ECHO is set input characters are echoed back to the terminal.
- icanon: Canonical input (erase and kill processing). If ICANON is set canonical processing is enabled. In canonical mode input processing 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.
- onlcr: Map NL to CR-NL on output. If ONLCR is set the NL character is transmitted as the CR-NL character pair.

Sending data can be simply done by opening the device as a file and writing data to it.

- Configure a port on ttySTM1 (aka usart3). echo, icanon and onlcr properties are deactivated to handle raw data.

```
Board $> stty -F /dev/ttySTM1 115200 -echo -icanon -onlcr
```

- Display the current configuration on ttySTM1 (usart3):

```
# display the configuration of uart3 (aka ttySTM1) #
Board $> stty -a -F /dev/ttySTM1
speed 115200 baud; rows 45; columns 169; line = 0;
```

- Open a port on ttySTM1 (usart3) to receive data

```
Board $> cat /dev/ttySTM1 &
```

- On the remote PC, identify the tty terminal associated to the RS232 card connected on STM32MPU USART3 pins

```
# Command to execute from host terminal #
PC $> ls /dev/ttyUSB*
/dev/ttyUSB0
```

- Open a minicom in a second terminal on the remote device connected on USART3 pins

```
PC $> minicom -D /dev/ttyUSB0
```

- Display the current configuration on ttyUSB0 (remote device):

```
# Display the configuration of host uart (aka ttyUSB0) #
PC $> stty -a -F /dev/ttyUSB0
speed 115200 baud; rows 45; columns 169; line = 0;
```

- Send data from remote PC to STM32MPU over USART3 with default termios configuration (8 frames length, no parity)

```
# Execute this command from host terminal #
PC $> echo "HELLO" > /dev/ttyUSB0
```

- Send data from STM32MPU to remote PC over USART3 with default termios configuration (8 frames length, no parity)

```
# Execute this command from STM32 terminal #
Board $> echo "HELLO" > /dev/ttySTM1
```

4.2 Parity errors detection

Some additional termios functions allow to enable parity errors detection:

- parenb: Parity enable
- parodd: Odd parity else even
- inpck: Enable input parity or framing check
- ignpar: Ignore characters with parity or framing errors

Exemples:

- Configure a port on ttySTM1 (usart3) with even parity enabling

```
# STM32 parity enabling #
Board $> stty -F /dev/ttySTM1 115200 -echo -icanon -onlcr parenb -parodd inpck ignpar
```

- Open a port on ttySTM1 (usart3) to receive data

```
Board $> cat /dev/ttySTM1 &
```

```
Open a minicom in a second terminal on the remote device connected on USART3 pins
PC $> minicom -D /dev/ttyUSB0
```

- Configure a port on ttyUSB0 (remote device) with even parity enabling:

```
# Remote device parity enabling #
PC $> stty -F /dev/ttyUSB0 115200 -echo -icanon -onlcr parenb -parodd inpck ignpar
```

- Send data from remote PC to STM32MPU over USART3

```
# Execute this command from host terminal #
PC $> echo "HELLO" > /dev/ttyUSB0
```

- Send data from STM32MPU to remote PC over USART3

```
#  Execute this command from STM32 terminal #
Board $> echo "HELLO" > /dev/ttySTM1
```

4.3 Framing errors detection

Some additional termios functions allow to enable framing errors detection:

- csize: Number of bits per byte (character size and parity bit configurations)
- inpck: Enable input framing check
- ignpar: Ignore characters with parity or framing errors

Exemples:

- Configure a port on ttySTM1 (usart3) with framing check enabling and 7 data bits length frames

```
# STM32 framing enabling #
Board $> stty -F /dev/ttySTM1 115200 -echo -icanon -onlcr cs7 inpck ignpar
```

- Open a port on ttySTM1 (usart3) to receive data

```
Board $> cat /dev/ttySTM1 &
```

```
Open a minicom in a second terminal on the remote device connected on USART3 pins
PC $> minicom -D /dev/ttyUSB0
```

- Configure a port on ttyUSB0 (remote device) with framing check enabling and 7 data bits length frames

```
# Remote device parity enabling #
PC $> stty -a -F /dev/ttyUSB0 115200 -echo -icanon -onlcr cs7 inpck ignpar
speed 115200 baud; rows 45; columns 169; line = 0;
```

- Send data from remote PC to STM32MPU over USART3

```
# Execute this command from host terminal #
PC $> echo "HELLO" > /dev/ttyUSB0
```

- Send data from STM32MPU to remote PC over USART3

```
# Execute this command from STM32 terminal #
Board $> echo "HELLO" > /dev/ttySTM1
```

5 Identify processes using a tty serial device (with fuser tool)

```
Board $> fuser /dev/ttySTM0
# The process numbered 395, 691 and 3872 are using a tty serial device #
395 691 3872
```

6 Link a tty serial device with a line discipline (with ldattach tool)

Attach ttySTM1 with line discipline number *n* :

```
Board $> ldattach n /dev/ttySTM1
```

7 File transfer over serial console

Please see the dedicated article How to transfer a file over serial console.

8 References

- stty manual page
- A Brief Introduction to termios: termios(3) and stty stty tutorial
- struct_termios man page

TeleTYpewriter

Microprocessor Unit

also known as

terminal input output structure