How to get Terminal

Contents

Stable: 26.09.2019 - 12:46 / Revision: 26.09.2019 - 12:44

Stable: 26.09.2019 - 12:40 / Revision: 26.09.2019 - 12:44

A quality version of this page, approved on *26 September 2019*, was based off this revision.

2 ways can be used to connect your PC to the board and then to control your board:

- through serial link (UART / USB)
- through network connection

Contents

1 Remote terminal via serial link (UART/USB)

First of all, you need to connect your host PC to the board via UART.

Depending on the board, some additional materials may be needed to physically interconnect the board to the console serial port (extension board or adapter to interconnect the UART from the board to the USB PC input).

Pictures describing board connections can be found in Category:ST boards

1.1 On Linux® PC

On Linux, we recommend to use *Minicom*.

You need first to check that Minicom is installed on your PC.

- From a Terminal window on your PC:

```
PC $> minicom
```

- If a message indicating Minicom does not exist, then **install** it:

```
PC $> sudo apt-get install minicom
```

- To **configure** Minicom:

The UART configuration that is set with the below command has to match the one from your board: to be checked within board user manual.

```
PC $> sudo minicom -s
```

Hardware flow control is generally set to off (default value is on) and a baud rate of 115200 is usual.

- To **launch** Minicom:

```
PC $> minicom -D /dev/tty...
```

See How to use TTY with User Terminal article to find the appropriate tty instance to be used.

See below example for a standard UART (ttyS0) or STLink interface (ttyACM0):

```
PC $> minicom -D /dev/ttyS0
PC $> minicom -D /dev/ttyACM0
```

On the terminal, the prompt is changed to:

```
root@stm32mp1:~#
```

More information on Minicom can be found at the following link: https://help.ubuntu.com/community/Minicom.

1.2 On Windows® PC

Any of the Windows terminal emulator applications can be used. "Tera Term" is one of them: http://tera-term.en.lo4d.com/.
Then, the configuration is quite simple:

- Plug the cable and start the board
- Open a terminal emulator application on your PC and configure with the serial setup menu the USB port to be used and the baud rate of the serial link (in general 115200 baud)
- More information on the board serial port can generally be found in the hardware user manual **of the board**

The setup is correct when you can send Linux commands through the terminal emulator and get feedback. You can for example reboot the board and see the boot sequence.

2 Remote terminal over network

The board can also be accessed via the network through Ethernet connection, using ssh.

To do so, first **get the IP address** of the board:

- if it exists, using the board user interface (Terminal window or specific application)
- using a console (see chapter Remote terminal via Serial link (UART/USB)

If the network connection is through Ethernet then "InterfaceNetwork" = "Eth0", else if the network connection is using Wifi then "InterfaceNetwork" = "WLAN0"

```
Board $> ifconfig InterfaceNetwork
eth0      Link encap:Ethernet  HWaddr xx:xx:xx:xx:xx:FB
          inet addr:xxx.xxx.xxx.xxx <ip address>  Bcast:xxx.xxx.xxx.xxx Mask:xxx.xxx.xxx.
xxx
          inet6 addr: fe80::280:e1ff:fe01:14fb/64 Scope:Link
```

Then, from another terminal on a remote PC connected to the same network, use the command described in How to perform ssh connection.

Universal Asynchronous Receiver/Transmitter

Linux® is a registered trademark of Linus Torvalds.

ST in-circuit debugger and programmer for the STM8 and STM32 microcontroller families (See ST-LINK for more details)

Frame Buffer (could be the Kernel framebuffer linked to the display, a GPU framebuffer, an imaging framebuffer...)

Stable: 17.06.2020 - 15:27 / Revision: 16.01.2020 - 13:36

A quality version of this page, approved on *17 June 2020*, was based off this revision.

This category groups together all articles related to any STMicroelectronics board.

Pages in category "ST boards"

The following 8 pages are in this category, out of 8 total.

- MB1230
- MB1262
- MB1263
- MB1272
- MB1379
- MB1407
- STM32MP157x-DKx - hardware description
- STM32MP157x-EV1 - hardware description

Stable: 19.06.2020 - 10:06 / Revision: 19.06.2020 - 10:00

A quality version of this page, approved on *19 June 2020*, was based off this revision.

1 Purpose

This article describes how to perform remote connection using the **ssh**[1] tool.

1.1 Perform ssh connection

On host PC, one can type:

If it is a first connection:

```
PC $> ssh root@10.48.1.172
The authenticity of host '10.48.1.172 (10.48.1.172)' can't be established.
ECDSA key fingerprint is a0:a2:a3:09:b4:99:b3:90:6a:d0:35:05:6e:37:d0:6e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.48.1.172' (ECDSA) to the list of known hosts.
root@(none):~#
```

Otherwise:

```
PC $>ssh root@10.48.1.172
root@(none):~#
```

2 References

- [1] ifconfig

Elliptic Curve Digital Signature Algorithm
Stable: 17.03.2020 - 14:36 / Revision: 25.02.2020 - 15:13

A quality version of this page, approved on *17 March 2020*, was based off this revision.

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

Linux® is a registered trademark of Linus Torvalds.

also known as

terminal input output structure

Stable: 20.12.2019 - 15:12 / Revision: 12.12.2019 - 12:53

A quality version of this page, approved on *20 December 2019*, was based off this revision.

This article describes the ST-LINK hardware probes.

Contents

1 Introduction

The **ST-LINK**[1] is an in-circuit debugger and programmer for the STM8 and STM32 microcontroller families.

ST-LINK is a USB device and has to be connected to a PC host. It can be either embedded on ST boards or provided as standalone dongle.

ST-LINK can support different debug protocols depending on ST-LINK hardware version and on its embedded firmware version:

- SWIM: debug protocol for STM8 microcontrollers
- SWD/JTAG: debug protocol for STM32 microcontrollers and microprocessors

and communication interfaces:

- UART
- I2C
- SPI
- CAN
- GPIO

2 Hardware versions

Several versions of ST-LINK exist: ST-LINK/V1[1], ST-LINK/V2[2], ST-LINK/V2-A, ST-LINK/V2-B, ST-LINK/V2-1 and STLINK-V3SET[3].

For details about the different versions, please refer to technical note[4] of ST-LINK derivatives.

To find out the ST-LINK version that is embedded in your ST board, refer to the Category:STM32 MPU boards page, and then select your hardware board.

3 Getting started

3.1 Installing the USB driver

Two USB drivers are associated to ST-LINK, depending of ST-LINK version: one for the debugger itself, and one for the serial communication port from (***ST-LINK/V2-1***).
The serial communication port uses standard CDC ACM USB Class, which is usually present by default on all PC operating systems. The USB driver for the debugger can differ depending of the PC operating system:

- **MS Windows®**

A driver must be installed before connecting ST-LINK to a Windows® 7, Windows® 7 8, or Windows® 10 PC via the USB.

The driver is automatically installed by the toolsets supporting ST-LINK. It is also available from www.st.com [5].

- **Linux®**

Users must be granted with rights for accessing ST-Link USB devices. Rules must then be added into /etc/udev/rules.d.

All information and files for installing the udev rules are provided in the **STSW-LINK007**[6] package available from www.st.com (see *stsw-link007\AllPlatforms\StlinkRulesFilesForLinux\Readme.txt* file).

- **Mac OS®**

No specific installation is required.

3.2 Connecting JTAG/SWD for debugging

- Embedded ST-LINK

A JTAG/SWD link is available from the USB link provided by the ST-LINK. The USB device is mounted on the host PC and ready to be used.

- Standalone ST-LINK

Pins are available on the ST-LINK to connect the JTAG/SWD signals. Refer to Hardware versions for connection details.

- **JTAG**: VCC, JTDI, JTMS, JCLK, JRCLK, JTDO, NRST and GDN signals must be connected to the JTAG/SWD connector.
- **SWD**: VCC, SWCLK, SWDIO, NRST, SWO and GND signals must be connected to the JTAG/SWD connector (on some ST-LINK hardware version, a dedicated SWD port can also be available).

3.3 Connecting UART port (from *ST-LINK/V2-1*)

- Embedded ST-LINK

A UART serial port is available from the USB link provided by the ST-LINK. The USB device is mounted on the host PC and ready to be used.

- Standalone ST-LINK

Pins are available on the ST-LINK to connect the Rx/Tx and GND signals. Refer to Hardware versions for connection details.

4 To go further

4.1 Updating the embedded firmware

All information are given in the STSW-LINK007[6] software package available from www.st.com.

Warning

The firmware update application embeds the latest firmware version, so do not hesitate to get the latest version of the executable STSW-LINK007[6] software package

4.2 How to bypass the embedded ST-LINK

To use a newest version of ST-LINK standalone probe or another debug probe, ST-LINK can be bypassed for ST boards that already embed it.

- Deactivating the embedded ST-LINK

Please refer to the board description and schematic for how to put the embedded ST-LINK in reset mode if available.

- Bypassing the embedded ST-LINK signals and connecting an external hardware probe

Connect the relevant signals depending on the interface available on the new hardware probe. Please refer to Connecting JTAG /SWD for debug and Connecting UART port for standalone external ST-LINK.

5 References

- 1.01.1 https://www.st.com/en/development-tools/st-link.html
- https://www.st.com/en/development-tools/st-link-v2.html
- https://www.st.com/en/development-tools/stlink-v3set.html
- https://www.st.com/resource/en/technical_note/dm00290229.pdf
- https://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-utilities/stsw-link009.html
- 6.06.16.2 https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-programmers/stsw-link007.html

Single Wire Interface Module (debug protocol for STM8 microcontrollers)

Serial Wire Debug

debug and test protocol, named from the Joint Test Action Group that developed it

Universal Asynchronous Receiver/Transmitter

Inter-Integrated Circuit (Bi-directional 2-wire bus standard for efficient inter-IC control.)

Serial Peripheral Interface

Controller Area Network (robust bus mainly used for automotive applications)

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Linux® is a registered trademark of Linus Torvalds.

spelling for older versions of STLink, ST in-circuit debugger and programmer for the STM8 and STM32 microcontroller families

Operating System