# Device Interface on Host Machine

CLI commands are supported on both Linux and Windows OS. However, there are some limitations on the same functionality in the Windows environment, which are highlighted in the respective sections.

## Device Port Identification on Linux OS

Generally, the host interface connected to the device takes the form of the device USB ports on Linux OS, /dev/ttyUSBn. Once connected to Talaria TWO EVB, the host detects 4 USB ports for the connected device.

Type in the Ls command in Linux to list the USB ports:

|  |
| --- |
| $ ls /dev/ttyUSB\* |

Text

Description automatically generated with medium confidence

Figure 3: USB ports

From the ports listed, identify the following ports:

1. /dev/ttyUSB2: Connected to the peripheral UART of Talaria TWO module
2. /dev/ttyUSB3: Connected to the console port (GPIO17) which is unidirectional UART from the Talaria TWO module. It operates at a high baud rate 2457600 (default), used for debug prints.

With this information available, all CLI commands should use the following argument for the peripheral port to the device in the command line:

|  |
| --- |
| ./script/boot.py –-device /dev/ttyUSB2 --reset=evk42 apps/helloworld/bin/helloworld.elf |

This loads the helloworld app provided in SDK to Talaria TWO memory.

Launch miniterm, a small terminal application, to open the /dev/ttyUSB3 device port.

|  |
| --- |
| $ miniterm /dev/ttyUSB3 2457600 |

When an application runs on the device, it prints out debug messages through the console port, as shown in Figure 4.

Text

Description automatically generated

Figure 4: Debug messages displayed through console port

If there are multiple Talaria TWO devices where each of them has their own console port connected to the host via the USB interface, it will be sufficient to use the respective /dev/ttyUSBn with the command.

For instance, two Talaria TWO devices with USB connected.

Text

Description automatically generated

Figure 5: 2 Talaria TWO devices with USB connected

Figure 6 and Figure 7 show that the miniterm app is run on two separate terminals using respective device ports to communicate with the respective device.

Text

Description automatically generated

Figure 6: /dev/ttyUSB3

Text

Description automatically generated

Figure 7: /dev/ttyUSB7

## Device Port Identification on Windows OS

On a Windows platform, some control functionalities associated with the CLI device-reset feature requires additional support provided by installation of the libusbK library.

There are two ways for a Windows host to interface with the device EVB connected to the host via USB interface:

1. A Windows system with the libusbK installed (recommend)

The library can be installed with the Zadig tool: <https://zadig.akeo.ie/>

1. A Windows system without the libusbK installed

This is the default configuration when the EVB with USB is detected by Windows. It finds a generic, default USB/Serial driver to handle the device interface. In this setup, not all functionalities offered by the script tools are available.

Depending on the environment where the libusbK is installed or not, CLI use can be described as follows:

**On Windows Platform with libusbK installed (recommended)**

On this serial communication setup, Windows host is communicating with the device using the libusbK driver.

The libusbK’s installation and use by the host can be checked from the Windows Device Manager display as described in Figure 8.

Graphical user interface, text

Description automatically generated

Figure 8: libusbK driver installation

If there are multiple Talaria TWO devices connected, the Windows Device Manager appears as shown in :

Graphical user interface, text, application

Description automatically generated

Figure 9: Multiple Talaria TWO devices connected

With this setup in place, all CLI commands should use the following argument for the input port to the device in command line:

|  |
| --- |
| ftdi://ftdi:4232/3 |

where,

1. 4232: Device ID for the FTDI 4232H device on EVK
2. 3: FTDI UART interface ID for the input/programming on EVK

With this information available, all CLI commands should use the following argument for the peripheral port to the device in the command line:

|  |
| --- |
| python script\boot.py --device ftdi://ftdi:4232/3 --reset=evk42 apps\helloworld\bin\helloworld.elf |

This command loads the helloworld app provided in SDK to Talaria TWO memory.

If there are multiple Talaria TWO devices where each of them has their own peripheral port connected to the host via the USB interface, the device specifier must be expanded with the serial ID of the device in the command line. This is accomplished with the keyword --SN. Execute the following steps:

1. Append to the device port with the following string:

|  |
| --- |
| ftdi://ftdi:4232:**<device\_serial\_id>**/3, which is followed by: --SN **<device\_serial\_id>** |

1. Peripheral port to the device in the command line:

|  |
| --- |
| python script\boot.py --device ftdi://ftdi:4232:**1101-0391**/3 --reset=evk42 --SN **1101-0391** apps/helloworld/bin/helloworld.elf |

Text

Description automatically generated with medium confidence

Figure 10: Expanding the device specifier

<device\_serial\_id>:

The hyphen ‘-‘ is a valid character in a serial ID of a device, if present. In other words, a serial ID of 11010391 is considered different than 1101-0391.

Example of a device with serial ID 15100010:

|  |
| --- |
| python script\boot.py --device ftdi://ftdi:4232:**15100010**/3 --reset=evk42 --SN **15100010** apps/helloworld/bin/helloworld.elf |

Text

Description automatically generated

Figure 11:Device with serial ID 15100010

**On Windows Platform using COM port (without the libusbK installed)**

On this serial communication environment, Windows host is communicating with the device using the standard (default) USB/Serial driver.

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 12: Standard (default) USB/serial driver

In the listed ports, identify the following ports:

1. COM6: Connected to the peripheral UART of Talaria TWO module
2. COM7: Connected to the console port (GPIO17) which is unidirectional UART from Talaria TWO module and operates at a high baud rate 2457600 (default), used for debug print

With this setup in place, all CLI commands can use the following argument for the input port to the device in the command line:

|  |
| --- |
| python script\boot.py --device COM6 apps\gordon\bin\gordon.elf |

This command loads the gordon app provided in SDK to Talaria TWO memory.