The Library & The Tool

Welcome to libhush. The binaries, source code and documentation given hereby shall help you to integrate control of your hush device's operation states into your own scripts or software. The controllable states are:

- RED m!ka Arm OnAir signalling ON /OFF
- WHITE m!ka Arm OnAir signalling ON /OFF (in case you have a dual color OnAir indicator and your device is configured accordingly)
- MUTE state (only mutes mic signal with hush+ devices)

The package contains the following two major parts:

- the `libhush` hush library binary built for Microsoft Windows and macOS 64Bit operating systems
- the `toolhush` hush command line tool binary built for Microsoft Windows and macOS 64Bit operating systems & source code

For `libhush` you additionally will find a C/C++ header file. For `toolhush` you additionally will find the complete C source code and CMake build script.

The `toolhush` tool uses `libhush` and gives you command line options to get and set operation states of your hush device. The `libhush` library makes use of the none Yellowtec `libhidapi` project for USB HID support on different platforms. You will find `libhidapi` binaries built for Microsoft Windows and macOS 64Bit operating systems within this package as well.

The `toolhush` tool dynamically binds to the `libhush` library. The `libhush` library dynamically binds to the `libhidapi` library.

Whilst `toolhush` will be licensed under MIT and given to you with full source code hereby, `libhush` will be given to you as closed source binary only. For both components Yellowtec GmbH claims the Copyright.

The `libhidapi` is copyrighted by Alan Ott, Signal 11 Software, please see additional document `foss_info_lib-hidapi.pdf` for license information and authors.

> **NOTE:** On macOS simultaneous concurrent access to USB HID devices is not supported. As both the `libhush` hush library and the *Hush App* communicate to the device via USB HID you can only have one or the other running at a time but not both at the same time. In such situations, as the *Hush App* holds a long term connection to the device while the library just establishes a very short term connection, in most cases the library access will be the one to fail (and a call to it will return with error code of `2` aka. `LHUSH_RC_ERROR_DEVICE_NOT_AVAILABLE`).

The Binaries

All binaries will be found in the `/bin` folder.

The `/bin/macos` sub folder contains all the ones built for x86_64 & arm64 Apple macOS operating systems starting from macOS 10.13 aka. macOS High Sierra. The `/bin/windows` sub folder contains all the ones built for x86_64 Microsoft Windows operating systems starting from Windows 10.

The Sources

The source code and CMake build script for the `toolhush` tool will be found in the `/tool` folder.

Using The Tool

You can use the `bin/toolhush` tool binary *as-is* and incorporate a call to it from your shell or script of choice given the needed command line arguments to it.

```
USAGE: The tool supports three different actions: 'list', 'get' and 'set'.
        On 'list' lists all connected hush(+) devices with serial number and name.
        On 'get' gets current operation mode and states from a given device.
        On 'set' sets operation states on a given device

        'list' needs no further arguments:
                EXAMPLE: toolhush.exe list

        'get' supports 1 optional argument:
                1st: serial number of device to request mode and state on
                EXAMPLE1: toolhush.exe get 23HS000012
                EXAMPLE2: toolhush.exe get

        'set' supports 2 optional arguments:
                1st: states to enable as character codes
                2nd: serial number of the device to set state on.
                        - 'w' to enable white m!ka LED
                        - 'r' to enable red m!ka LED
                        - 'm' to enable mute (supported on hush+ only)
                EXAMPLE1 (white & mute ON): toolhush.exe set wm 23HS000012
                EXAMPLE2 (red ON, first found device): toolhush.exe set "r"
                EXAMPLE3 (all OFF, first found device): toolhush.exe set ""
                EXAMPLE4 (all OFF): toolhush.exe set "" 23HS000012
```

> **NOTE:** If you want to operate with more than one hush devices / have more than one hush devices connected to your host we strongly recommend using the serial number to specify the device you'd like to operate on. If not the 'first found' one will be used, but the device which will be the first found one may differ from run to run.

Using The Library

You can build your own software to dynamically bind and use the `libhush` library. Please checkout the source code file `tool/tool.c` of the tool as a reference or blueprint for how to use the supplied library functions.

The interface of the library is defined by the header file `tool/libhush.h`.

Build the tool

The binary of the `bin/toolhush` tool given to you hereby was built with CMake by the supplied `tool/CMakeLists.txt` script.

The following prerequirements have to be installed to build the tool as described here:

- CMake
- Visual Studio workload Desktop development with C++ (*if you want to build for Microsoft Windows*)
- Xcode & Xcode Command Line Tools (*if you want to build for Apple macOS*)

Build for macOS

Prepare

From Terminal run the following command to create the project files:

```
cmake -G "Xcode" -S ./tool -B build_tool_macos
```

- it is assumed the parent directory of the `tool` folder is your current directory
- `build_tool_macos` will be your project file's and build folder

Build

From Terminal run the following command to build:

```
cmake --build build_tool_macos --target ALL_BUILD --config Release
```

- on successfull operation the newly created binary `toolhush` will be copied to the `bin\macos` folder automatically

Build for windows

Prepare

From command line run the following command to create the project files:

```
cmake -G "Visual Studio 17 2022" -A x64 -S ./tool -B build_tool_windows
```

- it is assumed the parent directory of the `tool` folder is your current directory
- you might need to adapt the command to your currently installed version of Buildtools for Visual Studio and use another generator (the `-G` option)
- if cmake does not recognize the specified generator you maybe have to update cmake
- `build_tool_windows` will be your project file's and build folder

Build

From command line run the following command to build:

```
cmake --build build_tool_windows --target ALL_BUILD --config Release
```

- on successfull operation the newly created binary `toolhush.exe` will be copied to the `bin\windows` folder automatically