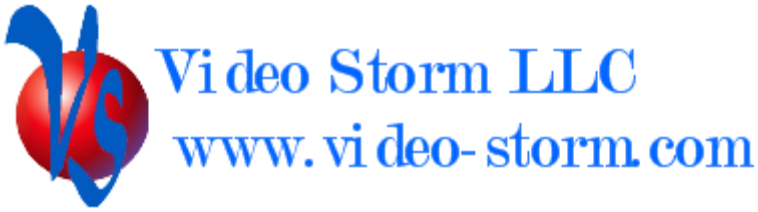

IrUSB Control guide

Revision 1.2
10/12/18

Overview

IrUSB exposes the following functionality to external APIs

- Device discovery
- IR TX (sending)
- IR RX (receiving)
- HID control (gui remote control of the attached device)
- General status and control

Hardware specifics

IrUSB uses a standard type A male USB port. The port is fully compliant with USB 2.0 device protocol. It can be connected to a USB host or hub. IrUSB is powered by the USB port as per USB specifications.

IrUSB features an IR transmitter located on the bottom side and an IR receiver located on the top side. It also features a visible LED indicator on the top side. This LED will flash when IR codes are transmitted.

Device drivers

Android or Android TV: Install the IrUSB app from Google Play or Amazon Appstore. The app contains a persistent service which will provide all control for the device. This app will also allow you to turn on/off your device using IR or TCP. The Android app allows for device setup via the app GUI or the TCP protocol.

Linux: Video Storm devices already have the required software installed. For 3rd party Linux devices, download the driver package from Video-Storm.com and follow the installation instructions.

API protocol settings

IrUSB TCP control port
Protocol: TCP/IP
Port: 9093 (multiple connection capable)

Device Discovery

IrUSB sends a multicast device discovery packet every 5 minutes. The format of this packet is as follows:

Address: 239.255.255.250
Port: 1904
Data: "Notify \nUUID \nIP_ADDRESS \nPORT \n0 \n\r"

UUID & IP_ADDRESS are set based on the network connected device (Android or Linux box). PORT is always 9093.

The QSTATVER command is used to query the number of attached IrUSB devices and their individual device_ids.

3rd Party Protocols

IrUSB currently supports the Global Cache 3rd party protocol for IR TX functionality. Device drivers for the iTach series should also work with IrUSB.

3rd Party protocols can work in conjunction with IrUSB protocol (they use different TCP ports).

Commands

IrUSB will echo back all commands sent to it. This is the easiest way to verify if your cable connection is correct. The NBX does not add <lf> after any <cr> received, so if you are using windows Hyper-terminal you should change the default settings to allow line feed on carriage return. IrUSB will only send the echo for valid commands terminated by <cr>.

All commands are terminated by <cr> (carriage return, ascii code 0xD). **NOTE: all references to <cr> in this document mean the single ascii character NOT the four characters "<cr>"**.

All IrUSB commands start with a Q and end with a <cr>

General command set:

QRESTART<cr>: Reboot IrUSB
QUPDATEFW<cr>: Updates the firmware reboots
QSDDPI<cr>: Send Identity multicast beacon
QWAKE<cr>: Wake device if sleeping
QSTATVER<cr> : Request device status

Output will be:
QSTATVER<cr>
OK<cr>
Driver/app Version string<cr>
IrUSB_device_version IrUSB_device_id<cr> (one line per attached IrUSB)

IR TX control:

QSIRPULSE ID=######## R=## hex_code <cr> Send hex code

The ID field is optional. It is the IrUSB_device_id to transmit this code. If ID is omitted all attached devices will transmit.

The R field is optional. It is the number of times to repeat the code.

The hex code should be proto hex format (captured/general IR format only).

Database commands (require configured IR codes)

Send preconfigured sink code
QDIRCODEyyy ID=######## R=##<cr> send predefined code yyy, repeat ##

Send preconfigured source code
QSIRCODEyyy ID=######## R=##<cr> send predefined code yyy, repeat ##

Send preconfigured named code
QSIRNCODE ID=######## R=## CODENAME<cr> send named code, repeat ##

IR RX:

When any attached IrUSB device receives an IR code, it will transmit this code to all connected TCP sockets using the following format:

QSIRPULSE000 hex_code<cr>

Database commands (require configured IR codes)
When any attached IrUSB receives a code matching a RX database code, it will send this event. NOTE: These are the ONLY events sent via the Cloud socket

QSIRNCODE CODENAME<cr>

External App Launching:

You can send Android intent URLs via the API to directly launch other apps on the connected device. The URL must be an android-app:// type url including the package and component names.

android-app://org.xbmc.kodi#Intent;component=org.xbmc.kod/.Splash;end

QLAUNCH URL<cr>

HID control:

Hid controls allow you to send keyboard or control keys directly to the attached box (Android/Linux) via the TCP API. This allows for external GUI or menu control or media transport controls.

QHIDCODEABBBCCC<cr>

A = 1 for keyboard codes short press, 2 for consumer codes short press
5 for keyboard codes long press, 6 for consumer codes long press
0 means cancel all keys (end any presses now)
BBB = Control keys for keyboard, Upper byte for consumer (decimal)
CCC = Key to send (decimal)

Code mapping is at:

https://source.android.com/devices/input/keyboard-devices#hid-keyboard-and-keypad-page-0x07

A few examples:

PLAY	QHIDCODE2000176
PAUSE	QHIDCODE1000072
HOME	QHIDCODE2002035
ENTER	QHIDCODE1000040
#	QHIDCODE1064032

Device metadata:

QGETPLAY<cr> Will return 0<cr> if no av playing, 1<cr> if av is playing
QGETFG<cr> Will return app_package<cr> which the package name of the current foreground app