Video Storm LLC
www.video-storm.com

IRUSB & usbMotion Control guide

Revision 2.1
8/8/23

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

usbMotion exposes the following functionality to external APIs
- Device discovery
- IR TX (sending)
- Motion triggers
- HID control (gui remote control of the attached device)
- General status, control, and configuration

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.

usbMotion 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, hub, or USB power supply. usbMotion is powered by the USB port as per USB specifications.

usbMotion features an IR transmitter located on the bottom side and an PIR (passive infrared) motion sensor 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 and otherwise indicates the state of motion detection. The LED can also be configured by our driver.

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.

usbMotion can also be used standalone in AUTO mode without a driver.

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 IRUSB 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)

        NOTE: IRUSB devices have version string starting with V2XX. usbMotion devices will return V3XX.
        usbMotion will also return the state of motion detect at the end of the line 0=no motion, 1=motion

IR TX control (both IRUSB and usbMotion):

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 (IRUSB only):

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>

Motion State & triggers (usbMotion only):

When any attached usbMotion device receives a motion trigger, it will transmit this code to all connected TCP sockets using the following format:

QMOTIONON devid<cr>
QMOTIONOFF devid<cr>

Devid is the 6 digit device id that has been triggered.

Motion state can be queried using the QSTATVER command above.

QMOTSET ID=######## a b<cr>
        Allow you to manual set the motion trigger state
        a = 0 or 1   motion state (set to)
        b = 0 or 1   If 1, then Motion triggers will fire if state changed
                        If 0, then no trigger will fire (state change only)

Motion Config (usbMotion only):

Device settings config: This allows setting the device configuration to control the filters on the motion sensor. These settings on stored in flash on the device.

QMOTCONFS ID=######## a b c d e<cr>
      a = On sensitivity, 0-31   (0 is longest range, 10 is recommended)
      b = Off sensitivity, 0-31   (0 is longest range, 10 is recommended)
      c = On filter, 0-63   (On state filtering, 2 is recommend)
      d = Off filter, 0-63   (Off state filtering, 5 is recommended)
      e = Config (1 is recommended for setup, 3 otherwise)

Sensitivity details: Lower values result in longer detection range. However, values less than 5 can also be subject to false triggering in some environments. When using lower values (most sensitive), we recommend also using non zero On and Off filters.

On filter details: 0 = disable all noise filters (only use this for debug)
                  1 = disable ON detection for 5 seconds after an OFF switch
                  2 = disable ON detection for 10 seconds after an OFF switch
                  3+ = as (2) above, but also require motion for X seconds before switching to the ON state. Note this will delay the trigger by X seconds.

Off filter details: 0 = disable all noise filters (only use this for debug)
                  1 = disable OFF detection for 5 seconds after an ON switch
                  2 = disable OFF detection for 10 seconds after an ON switch
                  3+ = as (2) above, but also require no motion for X-2 minutes before switching to the OFF state. Note this will delay the trigger by X-2 minutes.

LED config details: (bit field) Configures the functionality of the visible LED. Note that the visible LED can interfere with the motion detection at most sensitive levels. We recommend disabling the visible LED after installation.
                  (0) = disable OFF warning flash
                  (1) = disable ON led
                  (2) = LED is raw detect status
                  (3) = OFF led always flashes
                  (4) = reserved
                  (5) = reserved
                  (6) = Enable Toggle mode
                  (7) = disable TX led

Toggle mode changes the function to switch state on motion instead of the default state depends on motion detection.

Auto mode config: Auto mode allows usbMotion to automatically send HID or IR codes based on the ON and OFF motion triggers. Auto mode should only be configured when you will NOT be using the IRUSB app/driver for control. For example, to control power on/off on a PC (via USB HID) or on a TV (via IR only).

QMOTCONFHa ID=######## ABBBCCC<cr>
    a = code index, 0-3 are ON codes, 4-7 are OFF codes
    ABBBCCC is the HID code format from send hid command below

QMOTCONFIa ID=######## hex_code <cr>
    a = code index, 0=on 1=off
    The hex code should be proto hex format (captured/general IR format only).

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,
    3 for System command, 9 for Wakeup command,
    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
#           QHIDCODE1032032
WAKE        QHIDCODE9000000
SLEEP       QHIDCODE3002000
```

Keyboard mod keys: bit 0 is L CTRL, bit 1 is L SHIFT, bit 2 is L ALT, bit 3 is L GUI, bit 4 is R CTRL, bit 5 is R SHIFT, bit 6 is R ALT, and bit 7 is R GUI

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

Device search:

QHIDVS<cr>   Open up voice search on the device (microphone required)

QHIDTS searchtext<cr> Open up a text search with the given text