Author: Roman Leonov
email: roman.leonov@espressif.com

USB Host HID (Human Interface Device) Driver

Migration Guide from v.1.0.x to v2.0.x

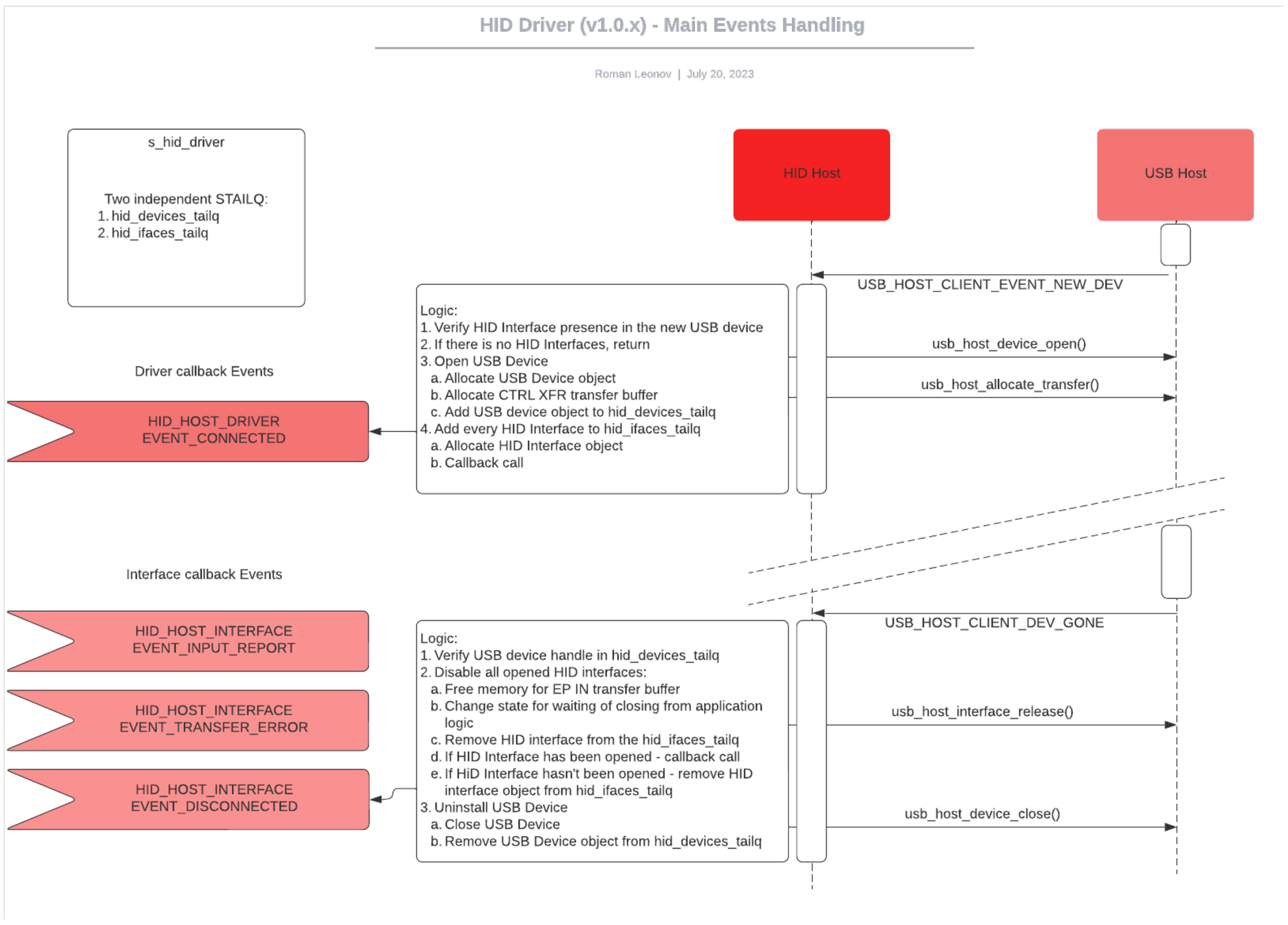
About

This document describes the architectural changes from v1.0.x to version v2.0.x. In the end of the document there are some tips to make the migration process from v1.0.x more easy.

Architecture Description

Driver v1.0.x

Main Events Handling


HID Driver (v1.0.x) - Main Events Handling
Roman Leonov | July 20, 2023
s_hid_driver
Two independent STAILQ:
1. hid_devices_tailq
2. hid_ifaces_tailq
HID Host
USB Host
USB_HOST_CLIENT_EVENT_NEW_DEV
usb_host_device_open()
usb_host_allocate_transfer()
Driver callback Events
HID_HOST_DRIVER EVENT_CONNECTED
Interface callback Events
HID_HOST_INTERFACE EVENT_INPUT_REPORT
HID_HOST_INTERFACE EVENT_TRANSFER_ERROR
HID_HOST_INTERFACE EVENT_DISCONNECTED
USB_HOST_CLIENT_DEV_GONE
usb_host_interface_release()
usb_host_device_close()

Logic:
1. Verify HID Interface presence in the new USB device
2. If there is no HID Interfaces, return
3. Open USB Device
   a. Allocate USB Device object
   b. Allocate CTRL XFR transfer buffer
   c. Add USB device object to hid_devices_tailq
4. Add every HID Interface to hid_ifaces_tailq
   a. Allocate HID Interface object
   b. Callback call

Logic:
1. Verify USB device handle in hid_devices_tailq
2. Disable all opened HID interfaces:
   a. Free memory for EP IN transfer buffer
   b. Change state for waiting of closing from application logic
   c. Remove HID interface from the hid_ifaces_tailq
   d. If HID Interface has been opened - callback call
   e. If HiD Interface hasn't been opened - remove HID interface object from hid_ifaces_tailq
3. Uninstall USB Device
   a. Close USB Device
   b. Remove USB Device object from hid_devices_tailq

Advantages

There is only one advantage:

1. After USB HID Device is connected and opened it is ready to work and be claimed from the application logic.

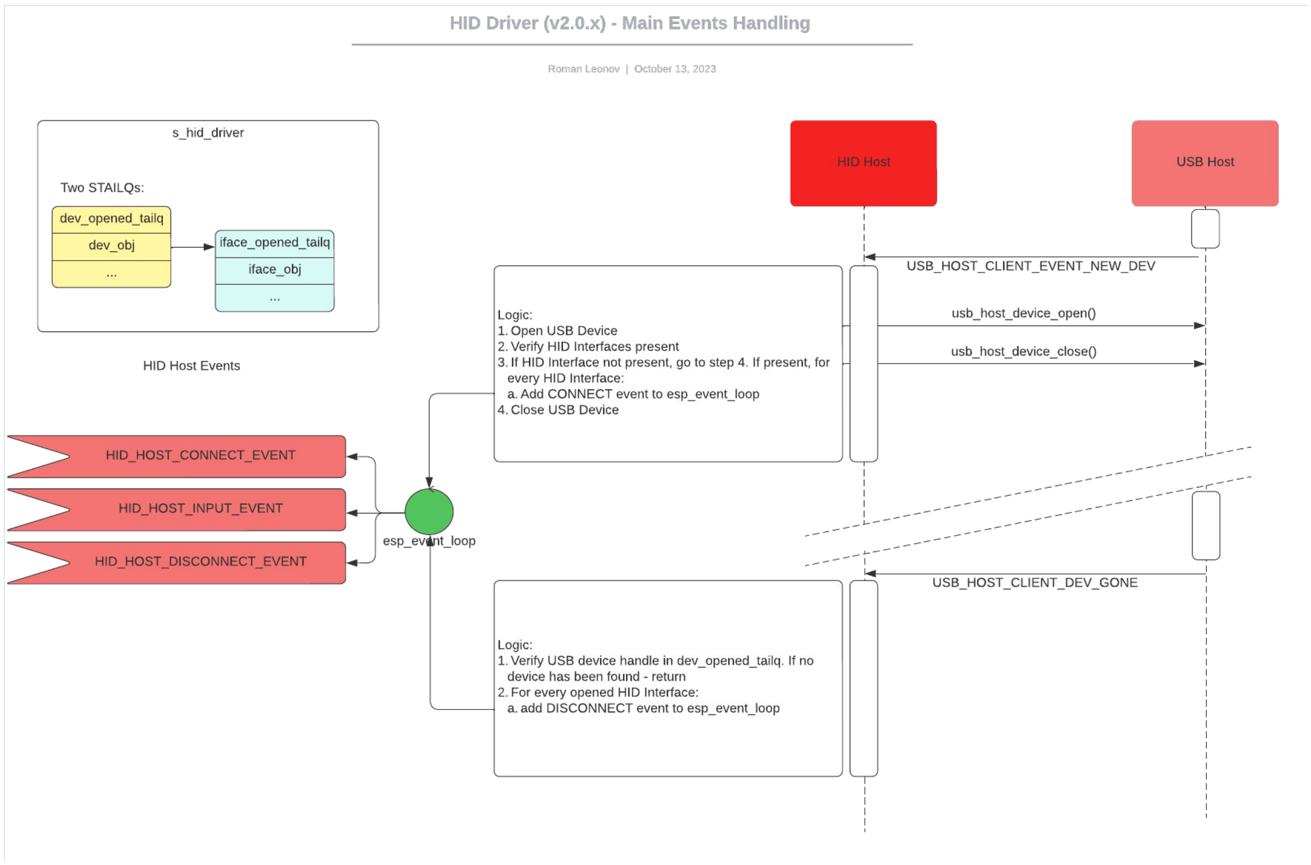
Disadvantages

There are several disadvantages:

1. Complex and heavy logic during event USB_HOST_CLIENT_EVENT_NEW_DEV (device creation, adding to the tail queue, memory allocation …);
2. Memory allocation was made even if the HID Device is not used in application logic;
3. Complex logic during device closing while device has been opened or hasn't been opened;
4. Complex and heavy logic during event USB_HOST_CLIENT_DEV_GONE (device removing, working with tail queues, …)
5. Requires to use the specific task in application logic to interact with HID device.

Driver v2.0.x

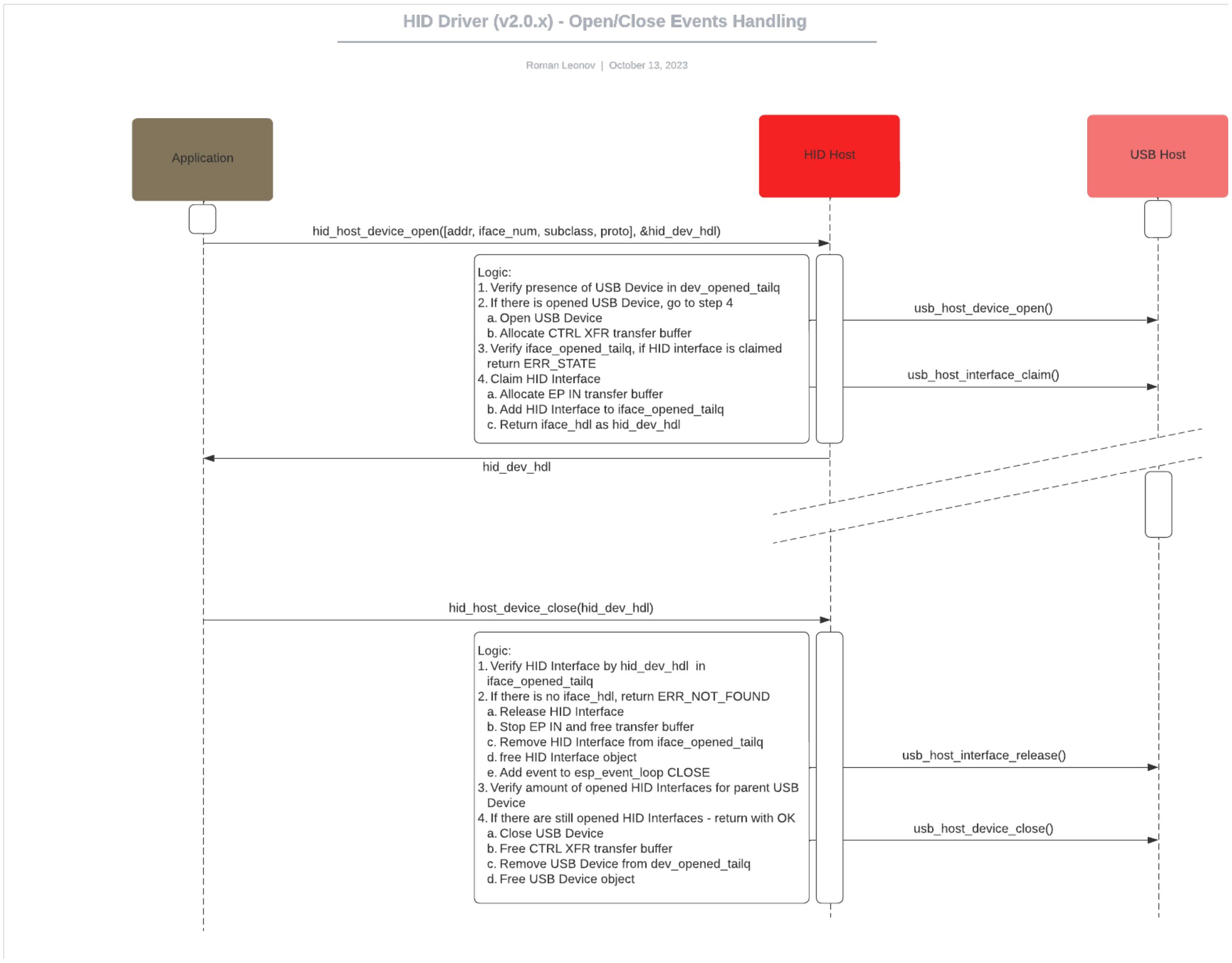
New Device/Dev Gone Events Handling



Logic:
1. Open USB Device
2. Verify HID Interfaces present
3. If HID Interface not present, go to step 4. If present, for every HID Interface:
   a. Add CONNECT event to esp_event_loop
4. Close USB Device

Logic:
1. Verify USB device handle in dev_opened_tailq. If no device has been found - return
2. For every opened HID Interface:
   a. add DISCONNECT event to esp_event_loop

Open/Close Events Handling


HID Driver (v2.0.x) - Open/Close Events Handling
Roman Leonov | October 13, 2023
Application
HID Host
USB Host
hid_host_device_open([addr, iface_num, subclass, proto], &hid_dev_hdl)
usb_host_device_open()
usb_host_interface_claim()
hid_dev_hdl
hid_host_device_close(hid_dev_hdl)
usb_host_interface_release()
usb_host_device_close()

Logic:
1. Verify presence of USB Device in dev_opened_tailq
2. If there is opened USB Device, go to step 4
   a. Open USB Device
   b. Allocate CTRL XFR transfer buffer
3. Verify iface_opened_tailq, if HID interface is claimed return ERR_STATE
4. Claim HID Interface
   a. Allocate EP IN transfer buffer
   b. Add HID Interface to iface_opened_tailq
   c. Return iface_hdl as hid_dev_hdl

Logic:
1. Verify HID Interface by hid_dev_hdl in iface_opened_tailq
2. If there is no iface_hdl, return ERR_NOT_FOUND
   a. Release HID Interface
   b. Stop EP IN and free transfer buffer
   c. Remove HID Interface from iface_opened_tailq
   d. free HID Interface object
   e. Add event to esp_event_loop CLOSE
3. Verify amount of opened HID Interfaces for parent USB Device
4. If there are still opened HID Interfaces - return with OK
   a. Close USB Device
   b. Free CTRL XFR transfer buffer
   c. Remove USB Device from dev_opened_tailq
   d. Free USB Device object

Advantages

There are several advantages:

1. No memory usage if HID device was not opened from application logic;
2. Event driven. There are only one callbacks to the application logic which are throwing via event_loop from HID Driver. No any additional logic during the callbacks from the levels underneath.

3. Simpler logic during device closing (we do not need keep the state of device in the driver).
4. Usage of event_loop let the application logic do not use specific thread and queue for HID Driver events handling.

Disadvantages

1. Usage of esp_event_loop inside the driver.
2. Breaking change in comparison with v1.0.x.

Migration Tips

API changes

TBD