ESP-Hosted Bluetooth

1 Bluetooth Stack

Currently, ESP-Hosted expects the Bluetooth stack needs to be run on host. So it would be worth to check the memory requirement for the preferred Bluetooth host stack are satisfied.

ESP-Hosted, functioning similar to relay, doesn't really limit or prefers any specific Bluetooth stack. Just because esp-nimble being readily available, esp-nimble is used to showcase the porting layer. Practically, users can use their own preferred Bluetooth stack with small porting effort.

As of current, ESP-Hosted uses esp-nimble, which is NimBLE stack readily available from ESP-IDF. esp-nimble is fork of Apache NimBLE. NimBLE stack provides Bluetooth Low Energy only (BLE only) functionality.

2 Bluetooth Controller

ESP-Hosted re-uses the Bluetooth controller running at slave. Slave is expected to be configured to use `controller only` mode.

As ESP-Hosted is just communication medium, it doesn't limit to BLE only. Classic BT stacks are also supported, given the slave has Classic-BT controller. The Classic-BT or BLE or both availability depends upon the Bluetooth stack support and ESP chipset chosen. As of today, ESP32 supports Classic-BT+BLE, whereas, the other ESP slave chipsets support BLE only.

3. Bluetooth Interface

Hosted allows two ways to use the Bluetooth stack running over host to communicate with the Bluetooth controller.

- vHCI
  - vHCI is standard HCI with extra headers or metadata
  - vHCI embedded **ESP-Hosted header** and re-uses the underlying **ESP-Hosted transport**, such as SPI/SDIO.
  - This option is much easier to set up. With this approach, once the existing SPI or SDIO transport is set up, in no time you get a working Bluetooth using vHCI driver.
  - When to prefer this option
    - Complete control of Bluetooth messages
    - Extra flexibility of debugging
    - No extra GPIOs for setting up (faster or no-set-up time)
- Standard HCI
  - Standard HCI is a transparent way of handling HCI messages
  - The pure HCI messages originated from Bluetooth stack running over Host, are sent through medium like UART to the Bluetooth controller at ESP
  - When to prefer this approach
    - Transparency - Messages not appended with headers.
    - Portability - Because of standard HCI, any slave is replaceable with any other co-processor chipset (ESP or any other as well)

3.1 NimBLE host stack with vHCI

ESP-Hosted currently works with the ESP-IDF NimBLE Bluetooth Stack through a VHCI driver.

The ESP-IDF NimBLE Bluetooth host expects the following APIs to init, and send Bluetooth Packets:

- `hci_drv_init`
- `ble_transport_ll_init`
- `ble_transport_to_ll_acl_impl`
- `ble_transport_to_ll_cmd_impl`

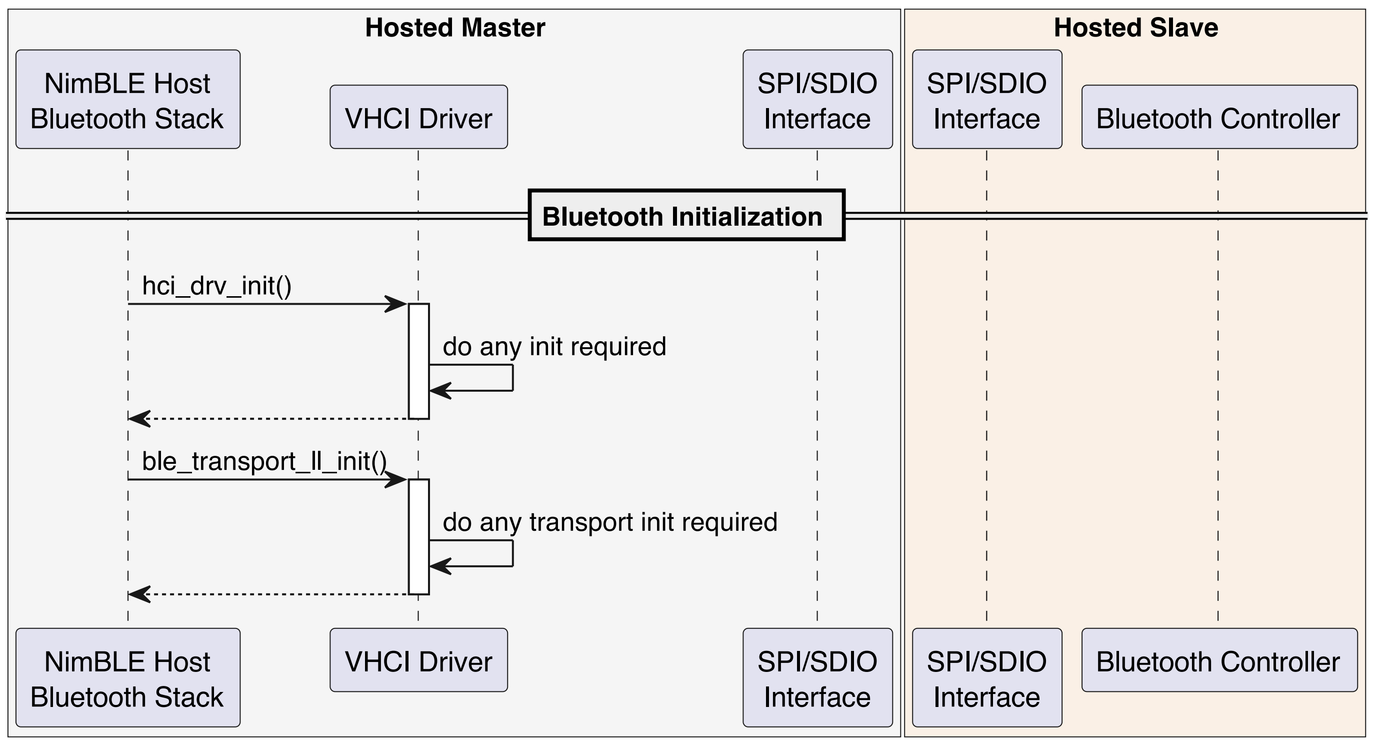
The ESP-IDF NimBLE Bluetooth host receives Bluetooth Event and ACL data through this interface:

- `hci_rx_handler`

3.1.1 Host vHCI Init

The following sequence diagrams show Bluetooth initialization and how the Bluetooth Host sends and receives data through Hosted.

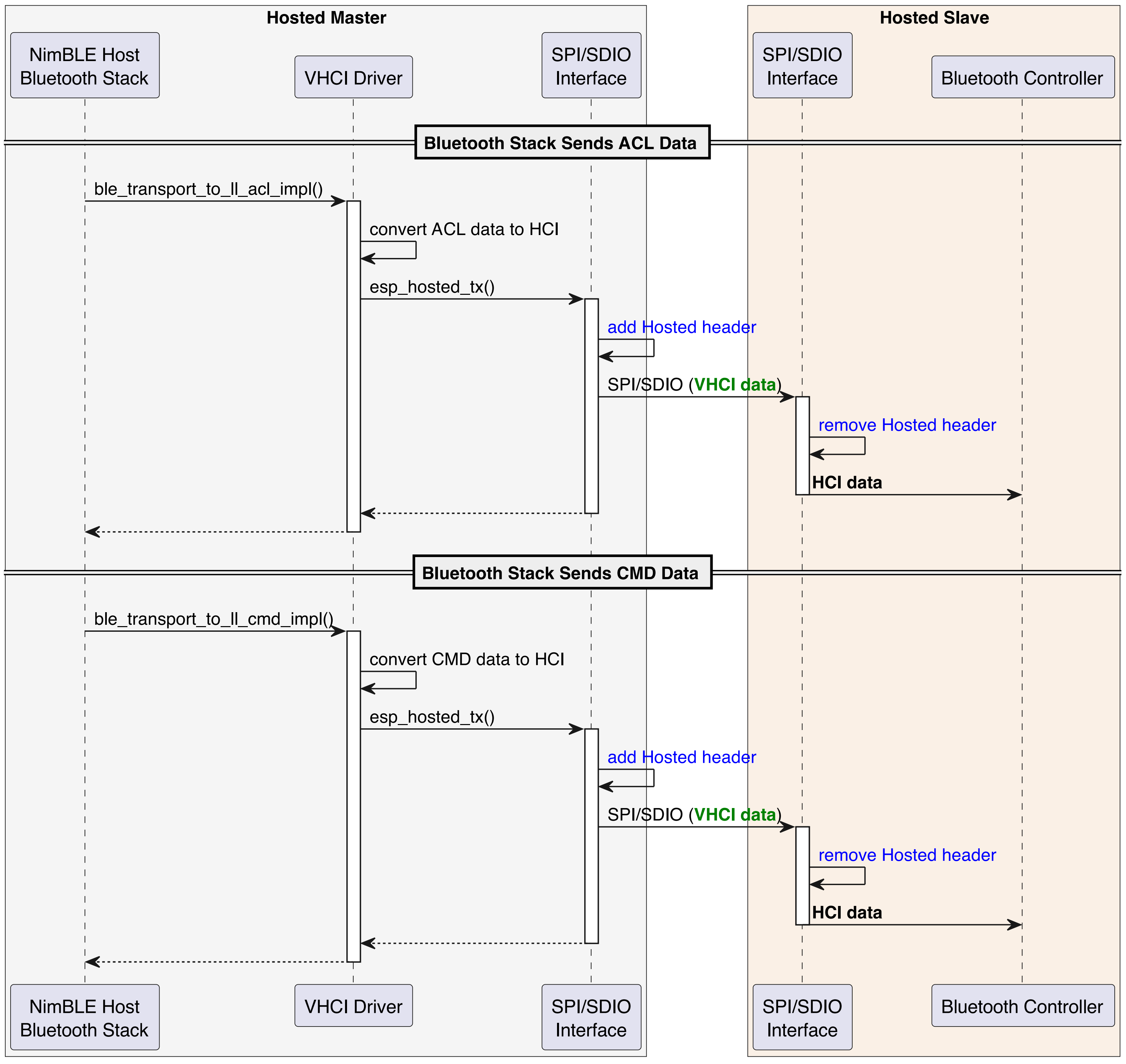
Current Bluetooth Implementation in ESP Hosted: Initialization



- PlantUML file for diagram: hosted_uart_init.txt

3.1.2 Host vHCI Tx

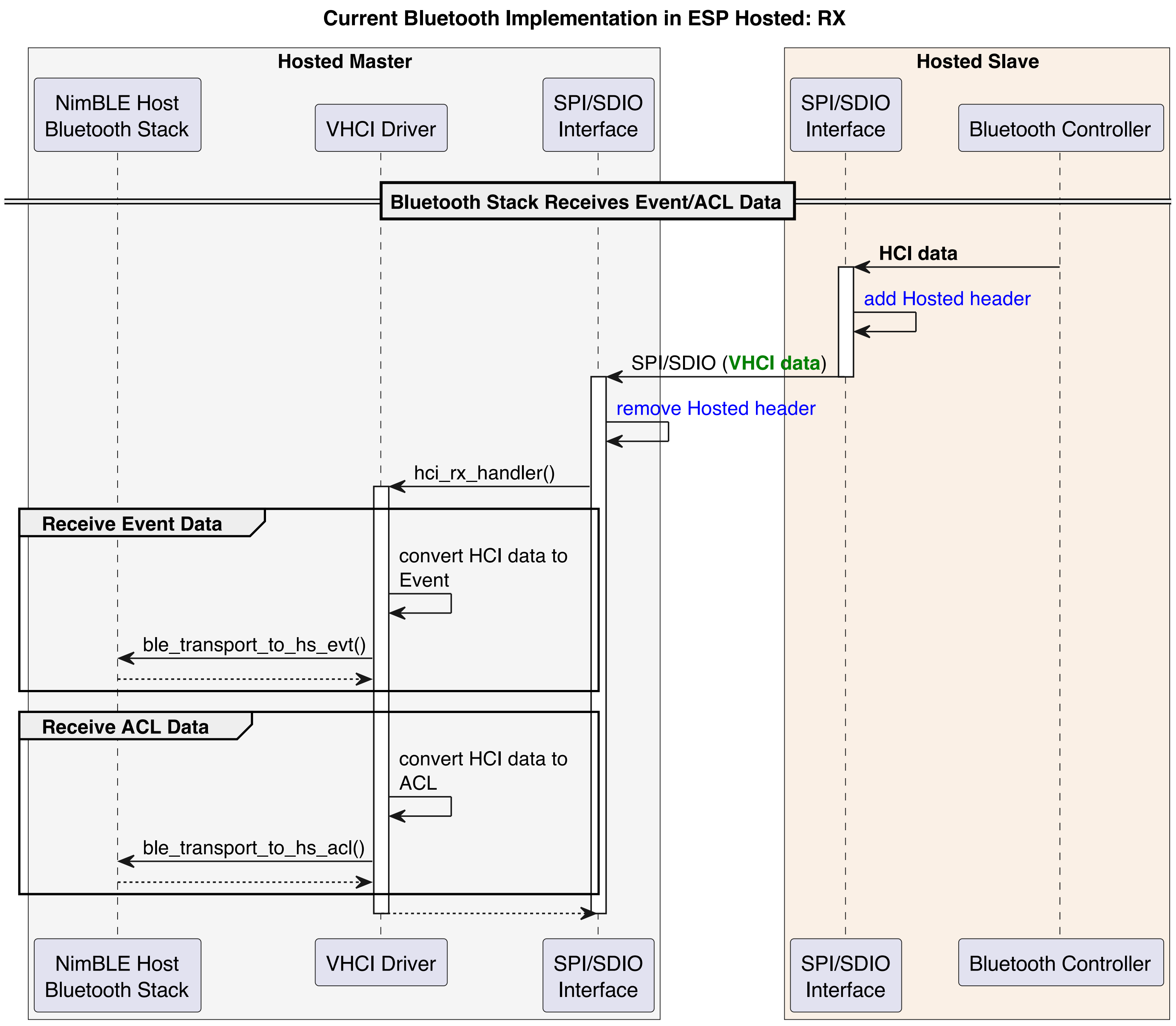
Current Bluetooth Implementation in ESP Hosted: TX



TX data from host Bluetooth stack to slave Bluetooth controller via vHCI

- PlantUML file for diagram: hosted_bluetooth_tx.txt

3.1.2 Host vHCI Rx



RX data from slave Bluetooth controller to host Bluetooth stack via vHCI

- PlantUML file for diagram: hosted_bluetooth_rx.txt

3.2 NimBLE host stack using standard HCI

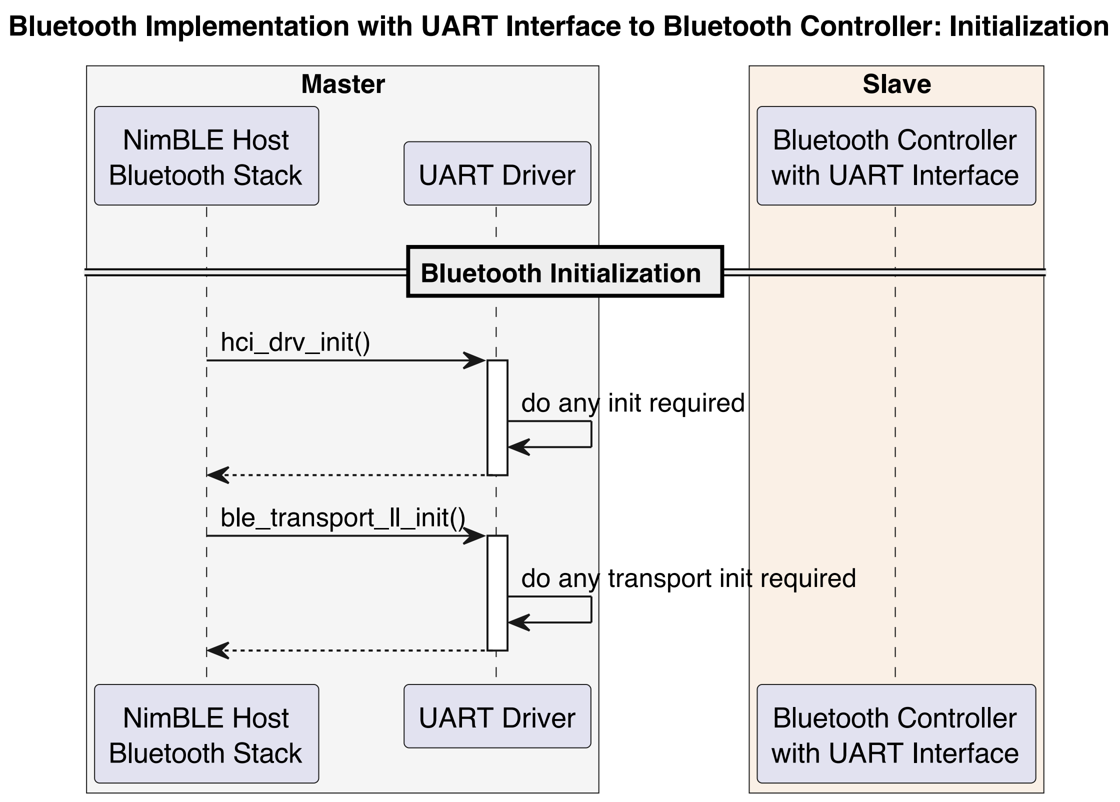
NimBLE also directly talk to a Bluetooth Controller through the UART interface. This does not involve any Hosted Code and only requires a UART driver.

Simple example of this configuration can be found in the following ESP-IDF NimBLE Host-only Example:

- on GitHub: [https://github.com/espressif/esp-idf/tree/master/examples/bluetooth/nimble/bleprph_host_only]

3.2.1 Host HCI Init

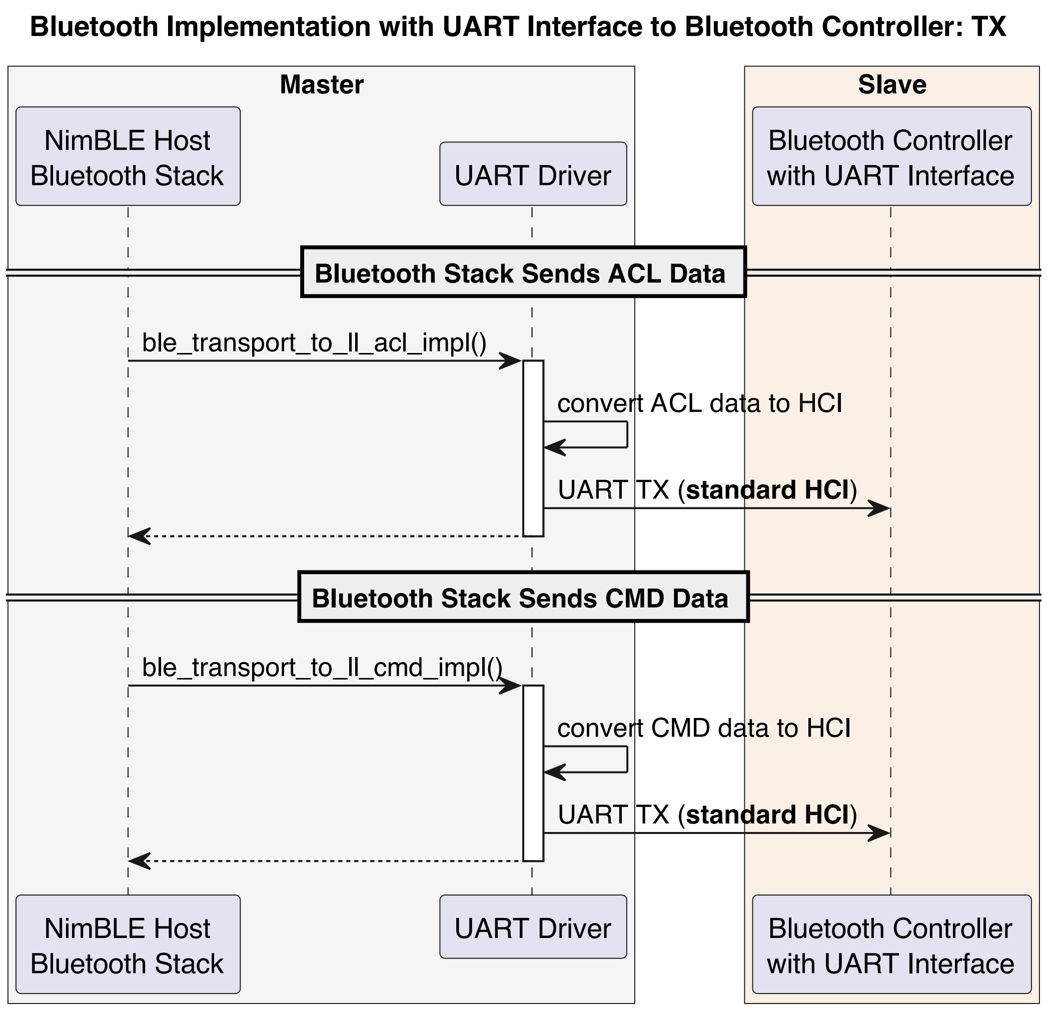
The following sequence diagram shows the Bluetooth initialization and how Bluetooth Host sends and receives data through UART.


Bluetooth Implementation with UART Interface to Bluetooth Controller: Initialization
Master
Slave
NimBLE Host Bluetooth Stack
UART Driver
Bluetooth Controller with UART Interface
Bluetooth Initialization
hci_drv_init()
do any init required
ble_transport_ll_init()
do any transport init required

Standard HCI over UART : Initialization

- PlantUML file for diagram: [hosted_uart_init.txt]

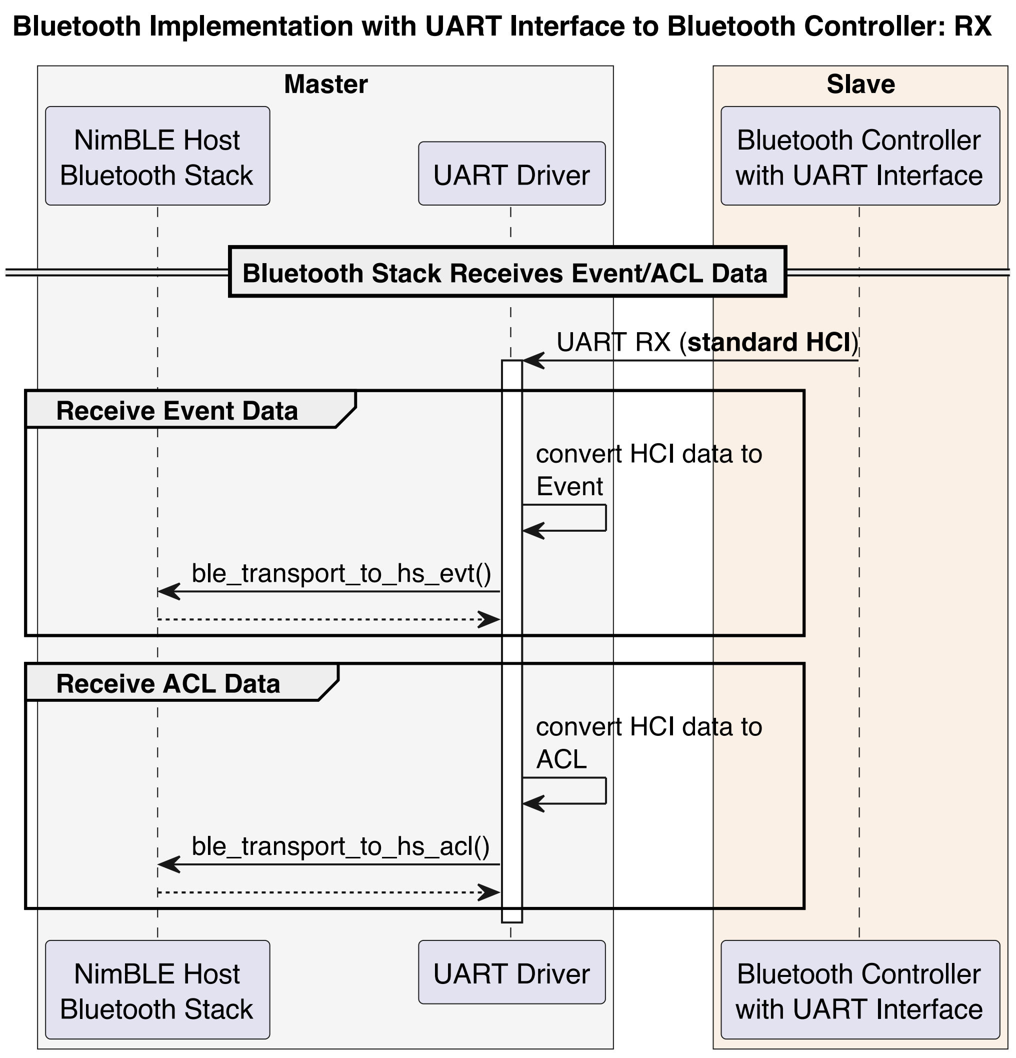
3.2.2 Host HCI Tx


Bluetooth Implementation with UART Interface to Bluetooth Controller: TX
Master
Slave
NimBLE Host Bluetooth Stack
UART Driver
Bluetooth Controller with UART Interface
Bluetooth Stack Sends ACL Data
ble_transport_to_ll_acl_impl()
convert ACL data to HCI
UART TX (standard HCI)
Bluetooth Stack Sends CMD Data
ble_transport_to_ll_cmd_impl()
convert CMD data to HCI
UART TX (standard HCI)

Standard HCI over UART : TX Data to Bluetooth Controller

- PlantUML file for diagram: [hosted_uart_tx.txt]

3.2.3 Host HCI Rx


Bluetooth Implementation with UART Interface to Bluetooth Controller: RX
Master
Slave
NimBLE Host Bluetooth Stack
UART Driver
Bluetooth Controller with UART Interface
Bluetooth Stack Receives Event/ACL Data
UART RX (standard HCI)
Receive Event Data
convert HCI data to Event
ble_transport_to_hs_evt()
Receive ACL Data
convert HCI data to ACL
ble_transport_to_hs_acl()

Standard HCI over UART : RX Data from Bluetooth Controller

- PlantUML file for diagram: [hosted_uart_rx.txt]