Host to Endpoint Sequence Diagram

(We combined our AHB slave and Data buffer to be one functional block)
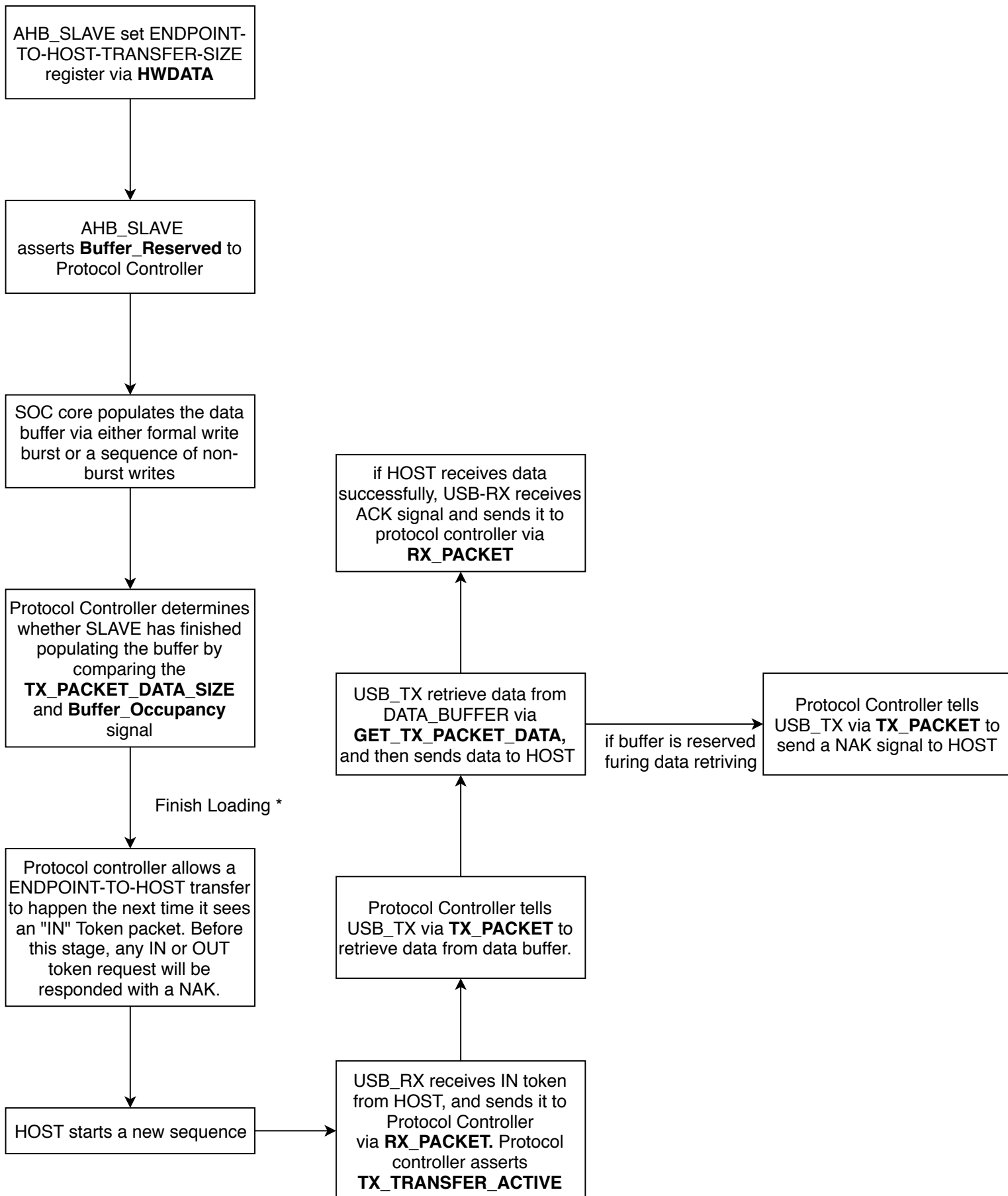
HOST starts New Sequence

USB_RX receives OUT token from HOST, and sends it to Protocol Controller via **RX_PACKET**

After HOST done sending data, Protocol Controller receives a done signal from USB_RX via **RX_PACKET.** It then tells USB_TX via **TX_PACKET** to send a NAK signal to HOST

Protocol Controller checks whether Data Buffer was empty via **Buffer_Occupancy.** It asserts **RX_TRANSFER_ACTIVE**

DATA_BUFFER not empty

DATA_BUFFER empty

Data not valid or not properly stored *

USB_RX sends a store signal to Data Buffer via **STORE_PACKET_DATA**

Data valid and properly stored

Once all data has been retrieved from the buffer, AHB_SLAVE will clear the data available flag in the status register

After the Host done sending data, USB_RX sends a DONE signal to Protocol Controller via **RX_PACKET**

Protocol Controller tells USB_TX via **TX_PACKET** to send a ACK signal to HOST**.** It also assert **RX_DATA_READY**

After AHB_SLAVE receives **RX_DATA_READY** signal, it checks how much data has been received

SOC core retrieve the data via either formal read bursts or via a sequence of non-burst reads

* Data not porperly stored or not valid means during data-receiving process, if
(1) buffer is reserved
(2) Buffer Occupancy greater than 64
(3) CRC checksum does not pass

* Everytime a NAK signal been sent out, Protocol Controller asserts **RX_ERROR** to the AHB_SLAVE

Endpoint to HOST Sequence Diagram
(We combined our AHB slave and Data buffer to be one functional block)

AHB_SLAVE set ENDPOINT-TO-HOST-TRANSFER-SIZE register via **HWDATA**

↓

AHB_SLAVE asserts **Buffer_Reserved** to Protocol Controller

↓

SOC core populates the data buffer via either formal write burst or a sequence of non-burst writes

↓

Protocol Controller determines whether SLAVE has finished populating the buffer by comparing the **TX_PACKET_DATA_SIZE** and **Buffer_Occupancy** signal

↓ Finish Loading *

Protocol controller allows a ENDPOINT-TO-HOST transfer to happen the next time it sees an "IN" Token packet. Before this stage, any IN or OUT token request will be responded with a NAK.

↓

HOST starts a new sequence →

USB_RX receives IN token from HOST, and sends it to Protocol Controller via **RX_PACKET.** Protocol controller asserts **TX_TRANSFER_ACTIVE**

↑

Protocol Controller tells USB_TX via **TX_PACKET** to retrieve data from data buffer.

↑

USB_TX retrieve data from DATA_BUFFER via **GET_TX_PACKET_DATA,** and then sends data to HOST

→ if buffer is reserved furing data retriving →

Protocol Controller tells USB_TX via **TX_PACKET** to send a NAK signal to HOST

↑

if HOST receives data successfully, USB-RX receives ACK signal and sends it to protocol controller via **RX_PACKET**

* Finish Loading means **TX_PACKET_DATA_SIZE** equal to **BUFFER_OCCUPANCY**
Everytime a NAK signal been sent out, Protocol Controller asserts **RX_ERROR** to the AHB_SLAVE