# HCI command and rx code discussion

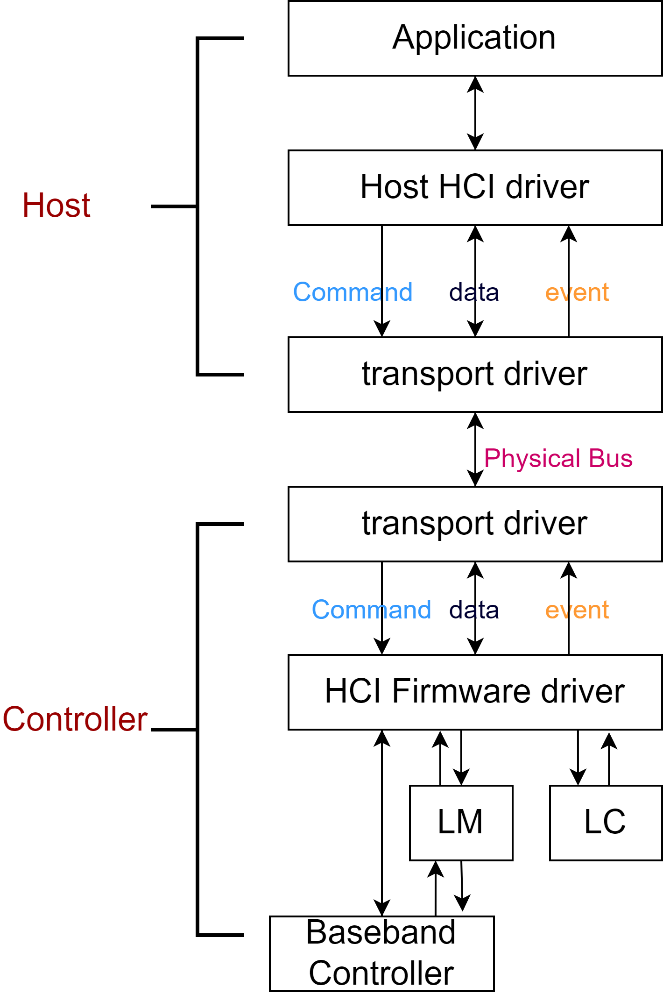
## HCI知识

HCI（Host Controller Interface）主机控制接口是蓝牙协议栈的一个重要部分。

HCI提供了对控制器的统一接口，作为Host和Controller之间的沟通桥梁。

HCI可分为三个部分，分别是

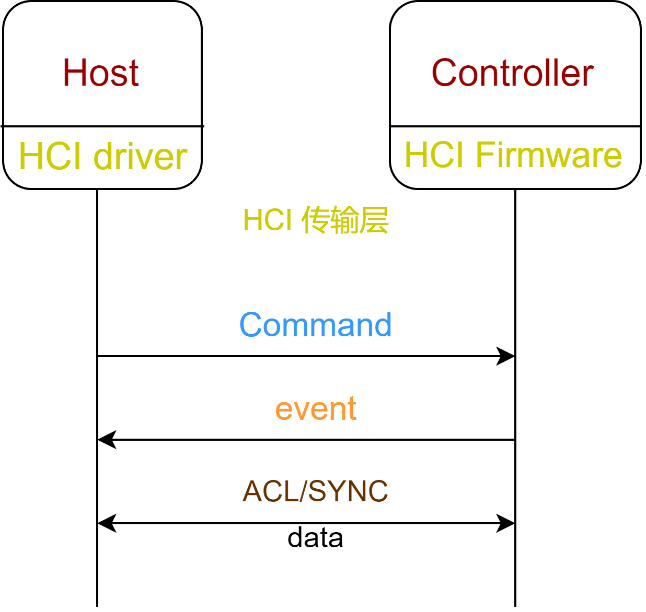
1. Host中的HCI driver：接收并处理HCI **events**
2. Controller中的HCI Firmware：访问HCI Commands和data
3. HCI传输层：进行**HCI Packet**的传输，在此主要介绍UART方式。



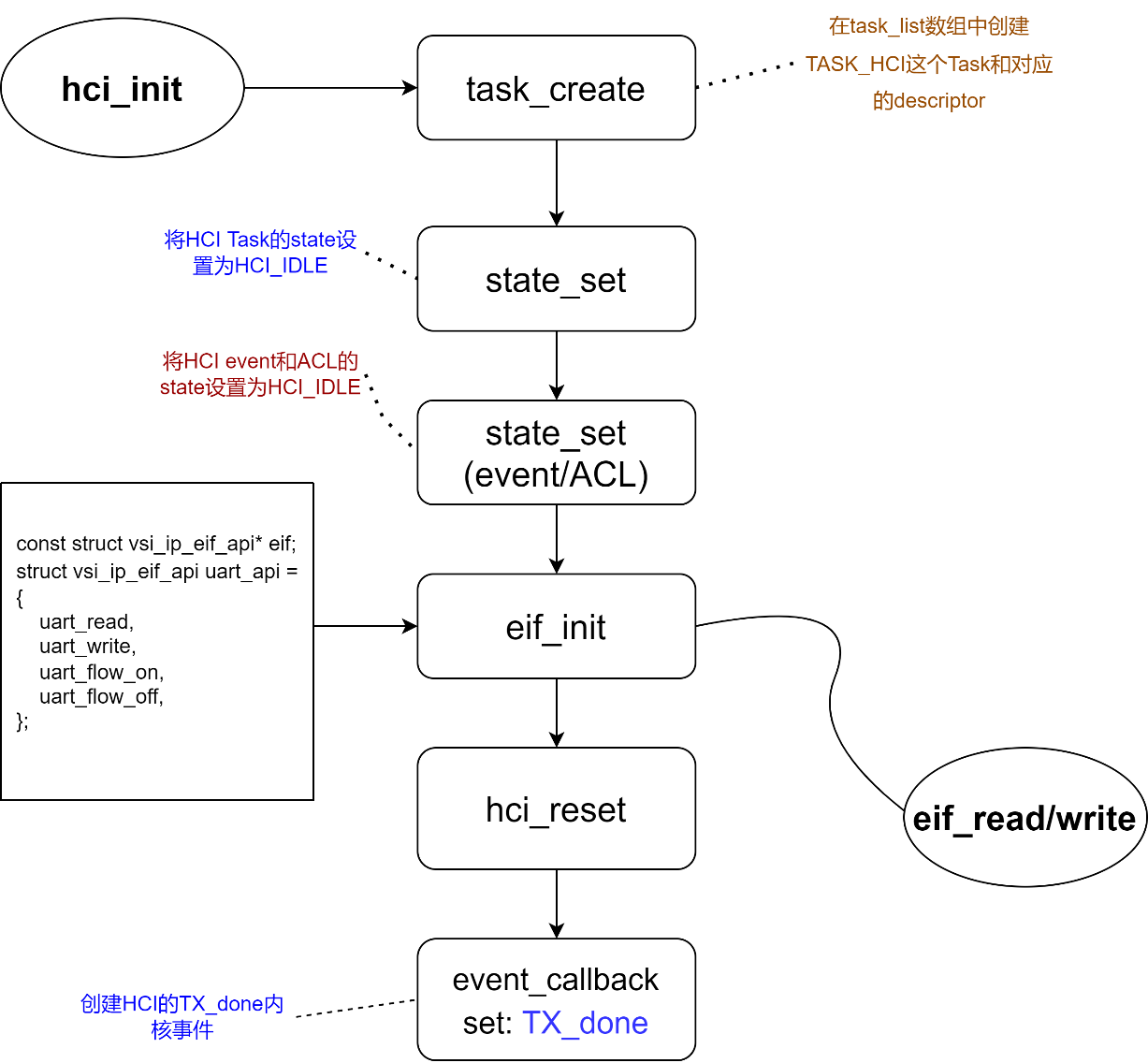
HCI通过包的形式来传送数据、命令和事件，

因此Packet分为三种类型：Command，Event，Data(ACL/SYNC)。其中Data是双向传输的，Command只能从Host发往Controller，而Event只能从Controller发送给Host。

由于HCI 不能直接区分四种 HCI 分组类型， 因此如果 HCI 分组通过一个通用物理接口（比如UART）发出， HCI 分组必须添加一个分组指示器Indicator。这个HCI分组指示器应在HCI分组之前立即发出，指示HCI的Packet类型。



## HCI的初始化

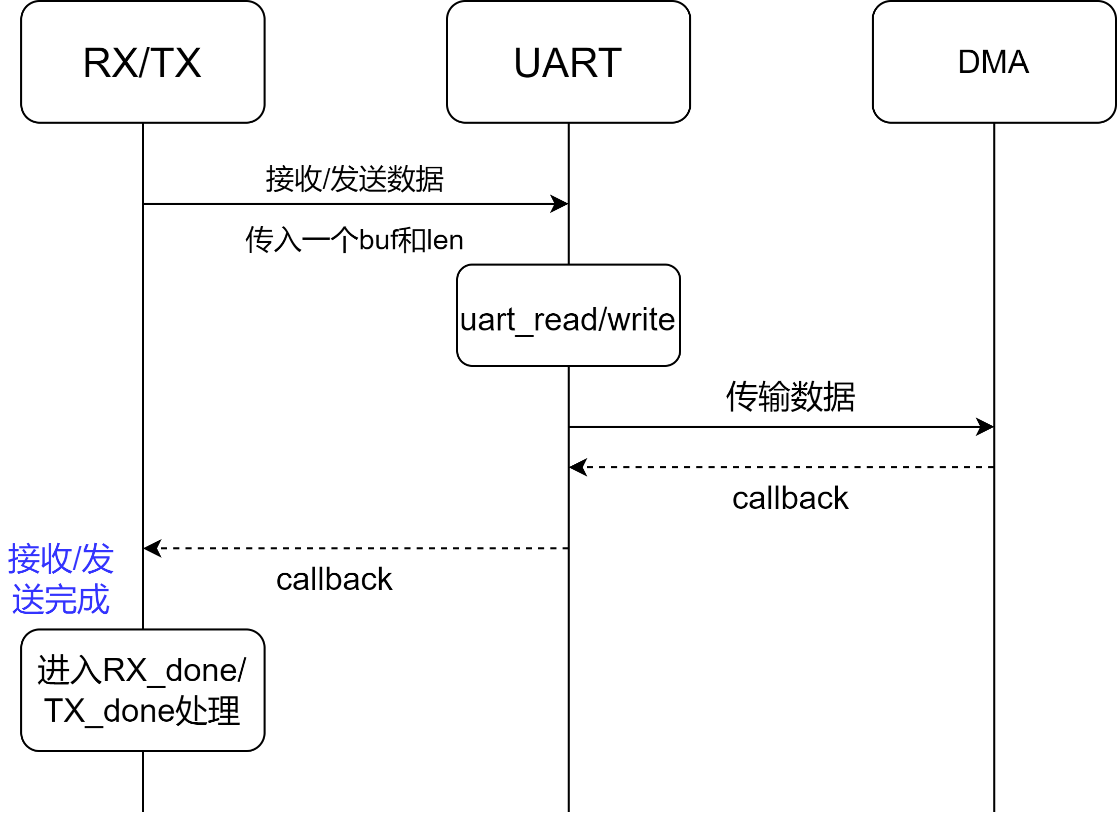


HCI在初始化时传入了一个结构体参数，指向uart\_api，即为uart\_read、uart\_write、uart\_flow\_on、uart\_flow\_off四个uart处理函数。而HCI的数据读取和发送均是通过uart完成的。

在初始化时，首先在内核中创建了一个TASK\_HCI的Task，并指定对应的descriptor。

接下来将其state设置为HCI\_IDLE，并进行hci的环境结构中evt\_queue和cal\_queue队列的初始化。之后初始化uart hci和hci的env中的curr\_payl\_buff、nb\_h2c\_cmd\_pkts，最后创建HCI的TX内核事件。

## 接收数据



在HCI接收数据时，数据是通过UART传输的。首先HCI接收的函数会调用uart\_read函数，并传入一个指针参数和长度参数，表示接收数据的保存位置和接收的数据byte。

之后会进入uart\_read，同时uart传输通过DMA来搬移数据，在数据的接收完成后回到RX\_done来处理接收到的数据。

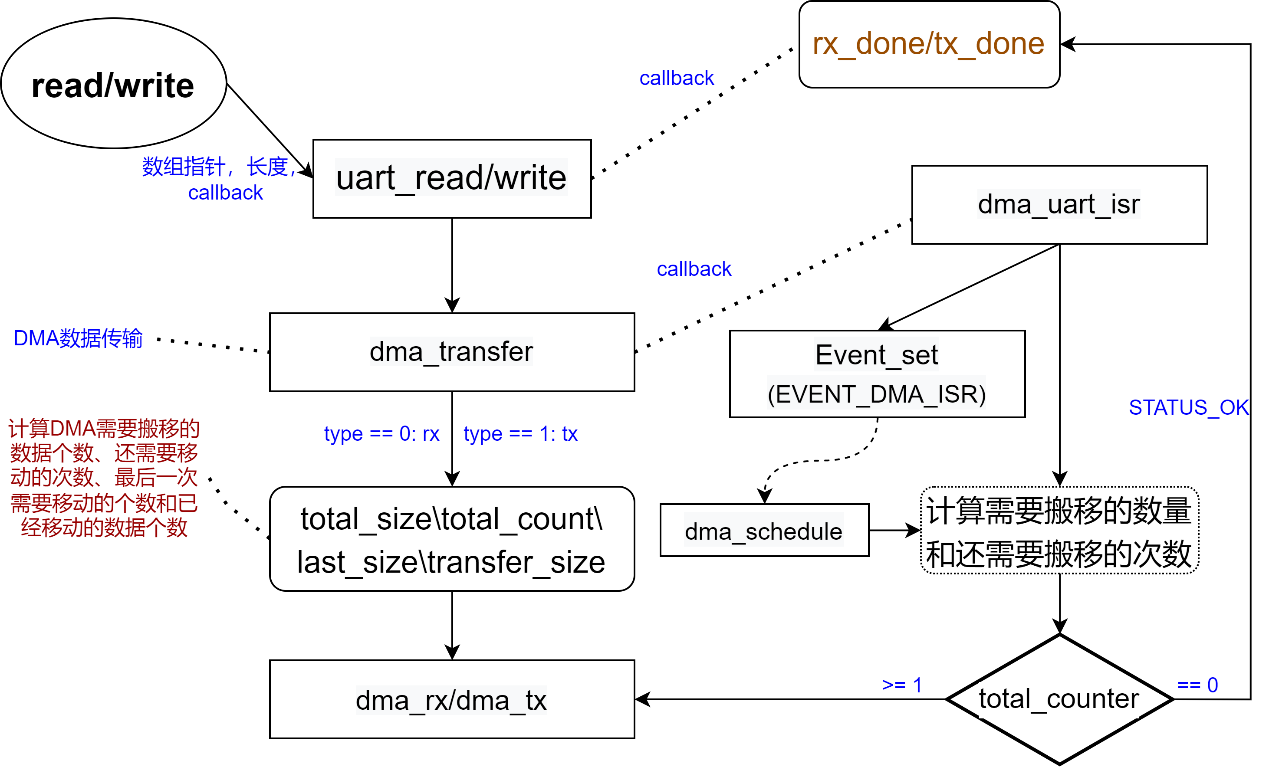
### UART

Uart模块完成HCI中发送、接收数据的行为功能。

在UART的函数中，会先检查传入的指针参数和读取byte的长度，在安全检查后进入DMA模块开始转移数据。

### DMA

DMA的搬移数据简单流程如下图：



DMA的传输会设置原地址（传入的数组指针）和目的地址、每次传输的数据byte等。

DMA的传输一次最多只能搬移31个数据，因此需要确定搬移的次数和每一次搬移的数据量，通过多次搬移完成数据的发送或者接收。

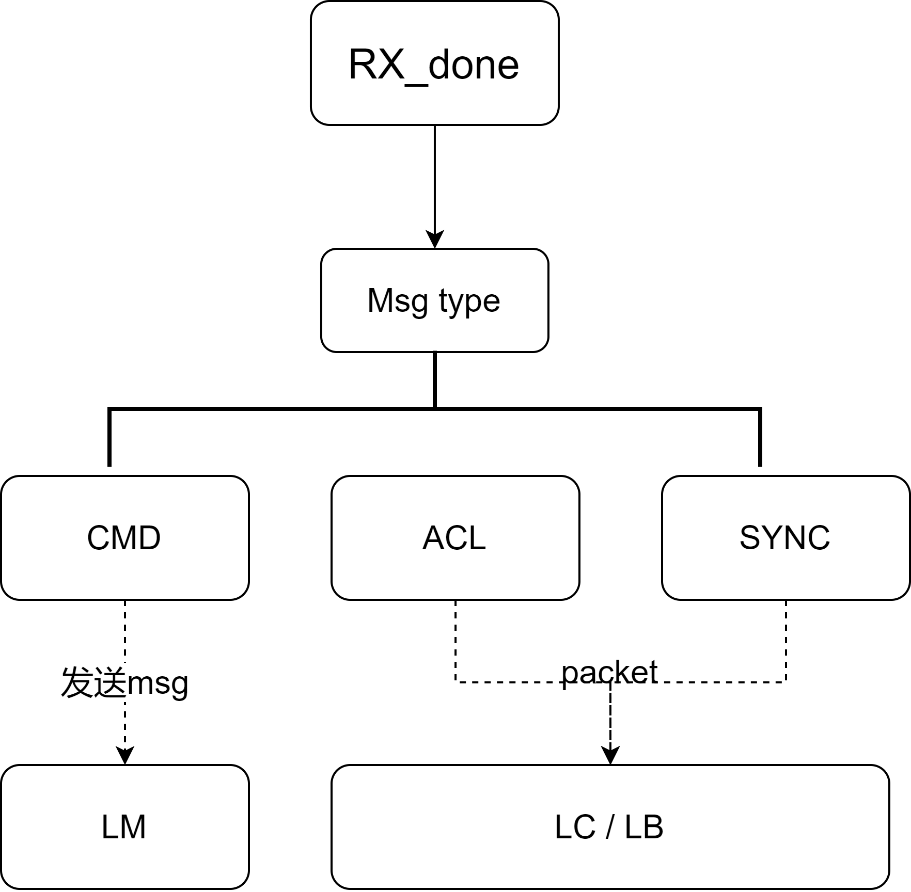
在确定了每次的搬移信息后进入dma\_rx/dma\_tx进行数据的传输。

DMA传输完成之后回到uart\_read/write的回调函数dma\_uart\_isr，在这里可以判断传输是否全部完成，未完成可以继续传输，完成后可以回到回调函数rx\_done中。当然，这个操作也可以通过内核消息KE\_EVENT\_DMA\_ISR的调度完成。最后uart\_read/write会返回rx/done中，返回的参数VSI\_IP\_EIF\_STATUS\_OK表示Status是OK的。

### HCI接收

#### 数据处理介绍

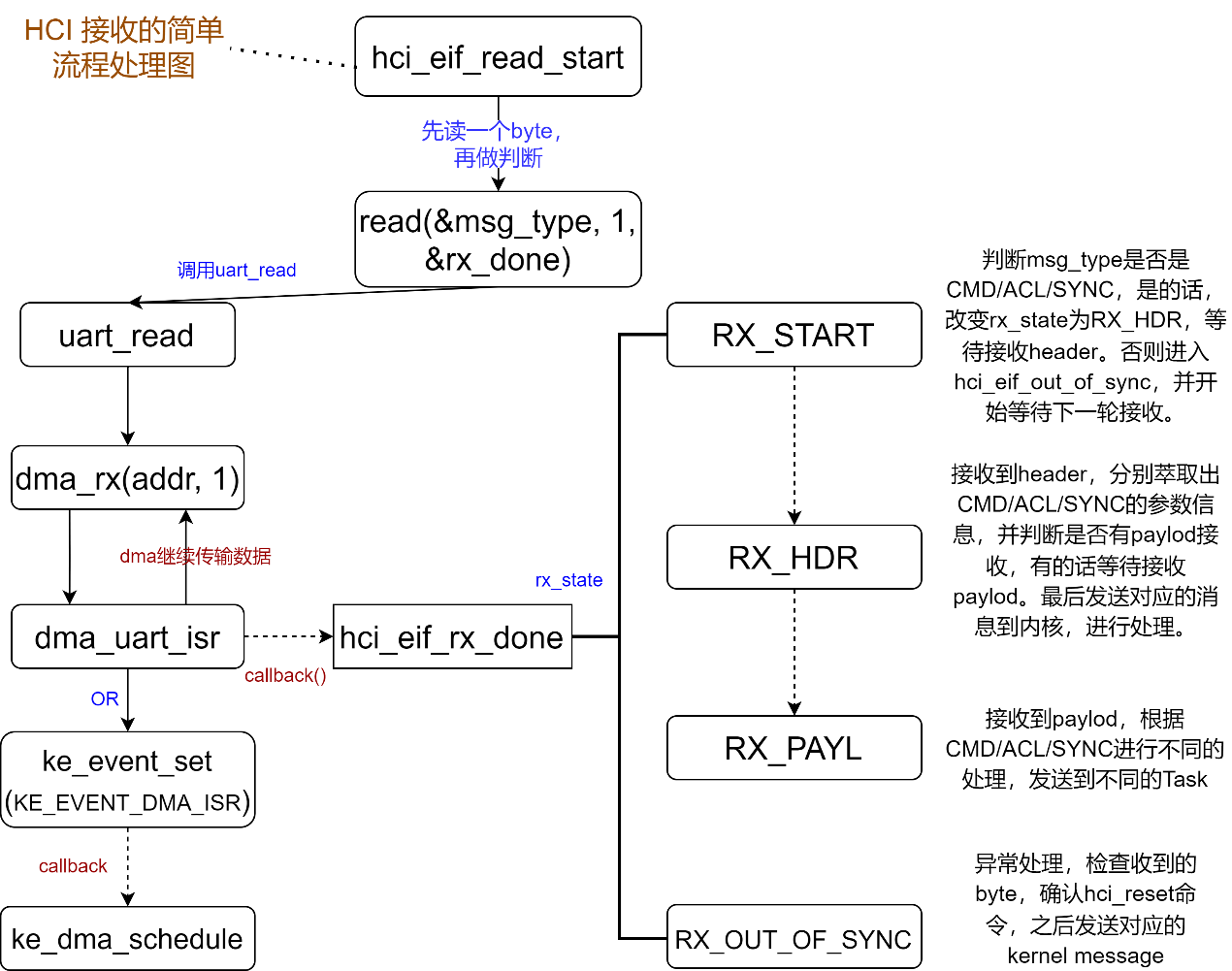
一个直观的理解：接收到了数据后会先判断类型，是Command就处理后发送给LM，是数据的话发送给Link Controller进行处理，是异常的数据也会进行异常的处理。



此时，一个简单的HCI接收数据后处理流程如下：

首先将tx的state设置为TX\_IDLE，即传输完成状态（而非等待传输状态），之后开始rx的读取数据。

Rx过程的开始首先将rx\_state设置为HCI\_STATE\_RX\_START，之后开始先读一个byte，读取数据包的indicator。根据这个indicator来判断当前读取到的包指向的是CMD\ACL\SYNC还是异常的数据。接下来的操作会根据这些分类有不同的处理。

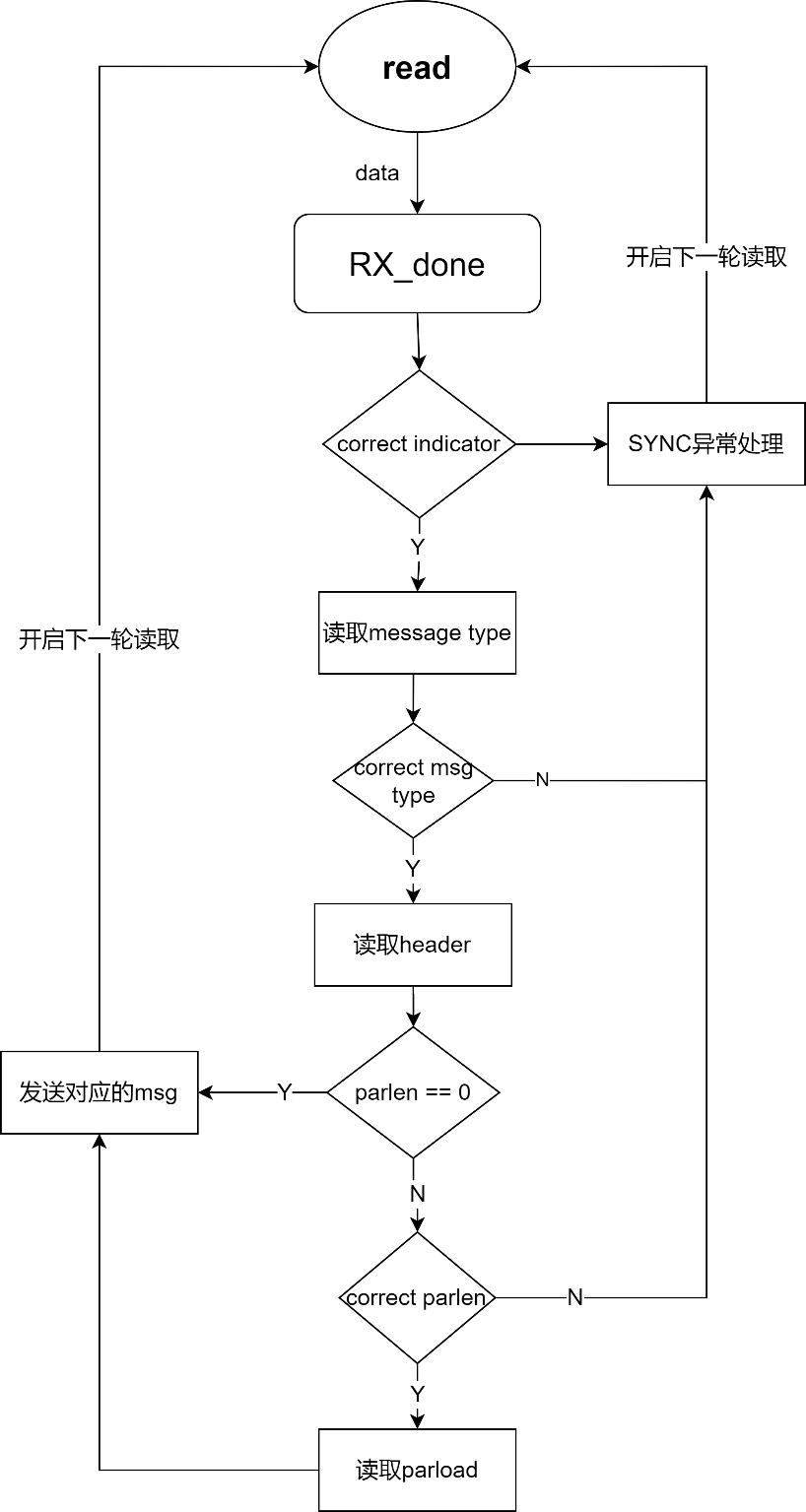


#### 2）数据接收流程

数据的接收遵循着这样一个简单流程：

接收indicator—>接收header—>接收payload—>处理并发送msg—>接收下一轮的indicator。

一个简单的数据接收流程图如下：



#### 3）接收数据的状态机

接收到的数据有四种情况：**包的indicator，包头header，包的参数payload，异常数据**。

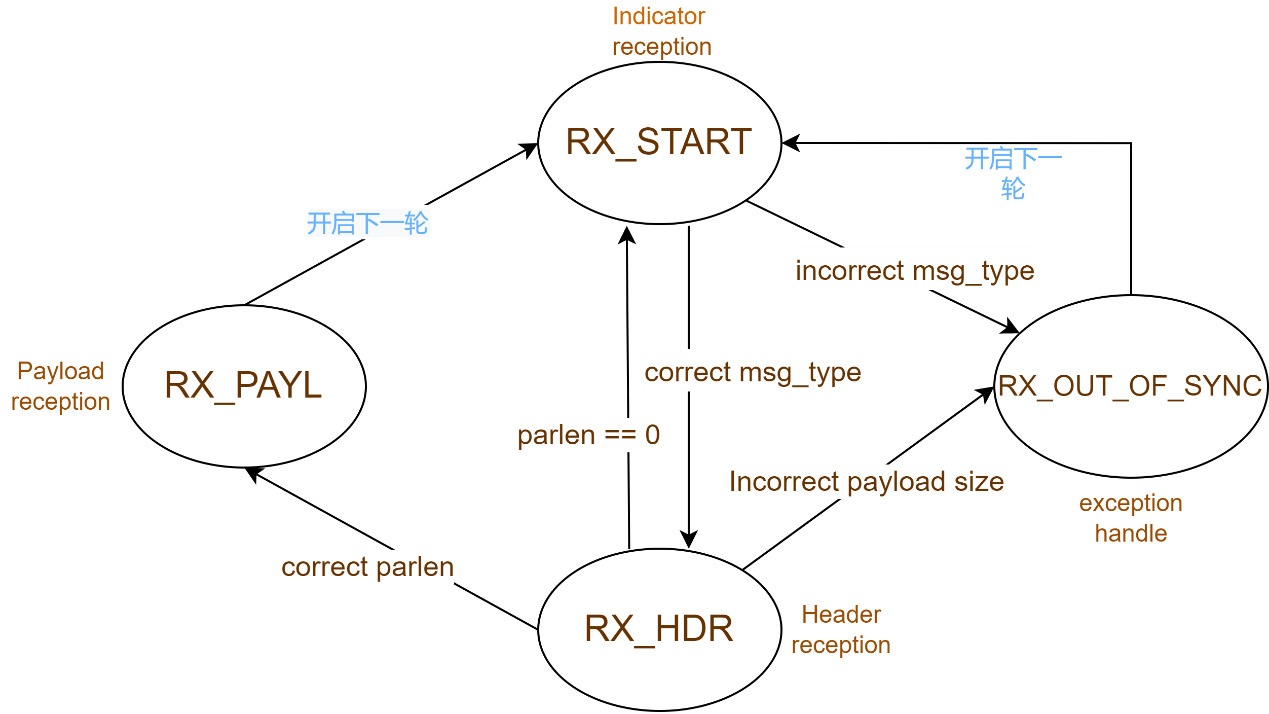
根据这四种情况分别有不同的处理。

**indicator**：判断当前indicator指向的包类型，根据不同类型接收接下来的header。

**header**：判断当前包类型，根据不同类型和包含的参数长度接收接下来的payload。

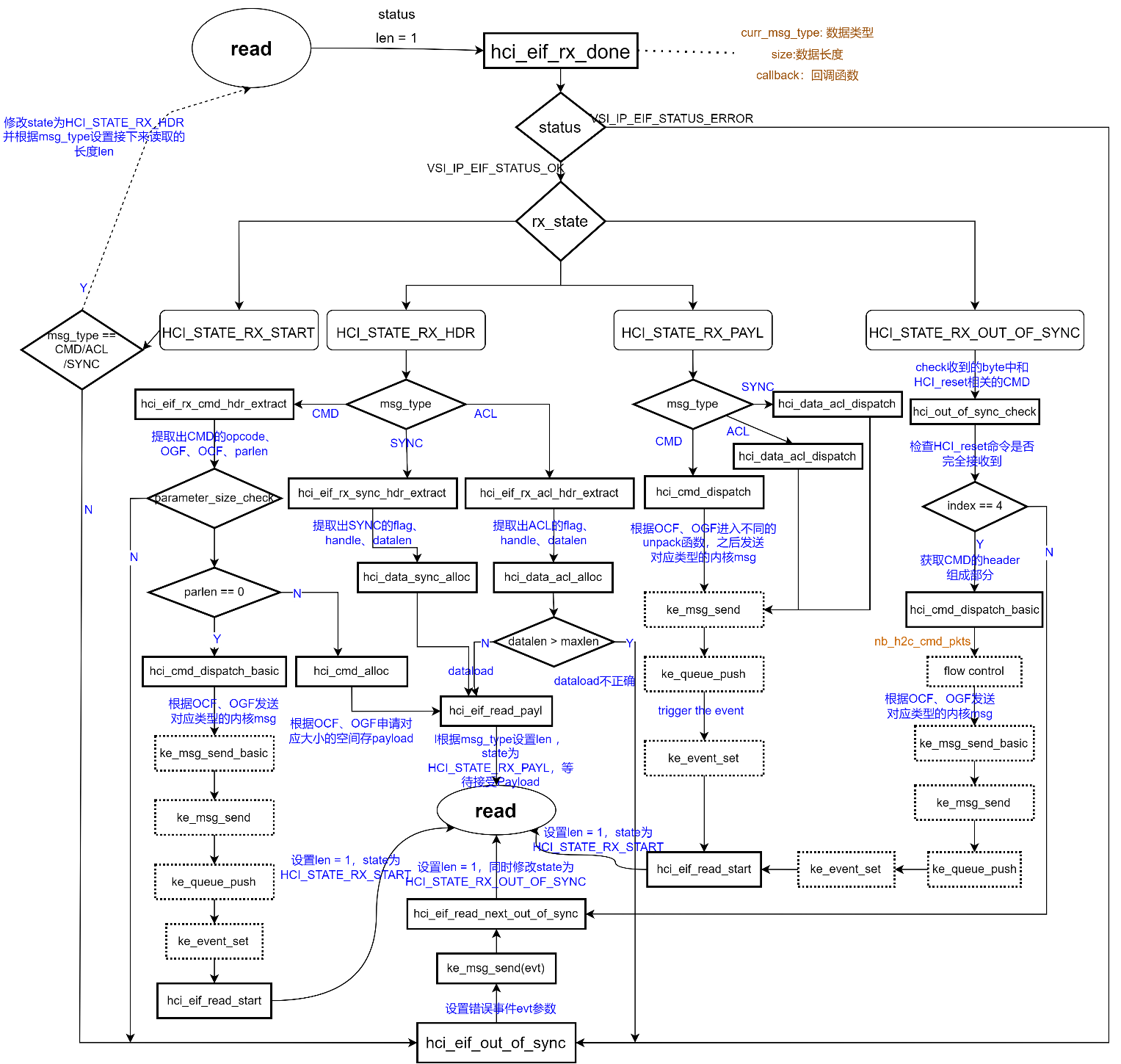
**payload**：判断类型，并进行对应的操作和处理。

**异常数据**：发送异常消息，开启下一轮的indicator读取。



#### 4）RX\_done的详细数据处理流程

而rx\_done的详细处理流程图如下：



**Rx\_done(uint8\_t status)**

为RX接收到数据后的处理handle。

在read\_start()中，设置接收到的信息的额外的接口环境。

根据接收到的数据类型：message，header，payload三种类型进行不同的操作。

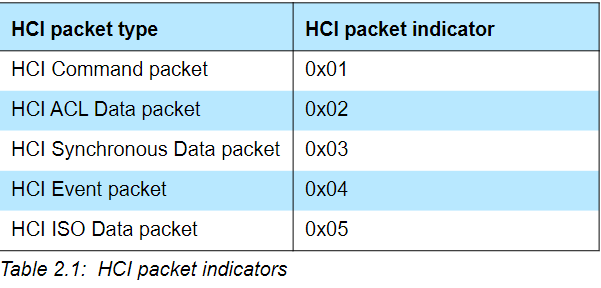
根据当前HCI的RX state来判断：

STATE\_RX有四种类型：

1. **HCI\_STATE\_RX\_START：等待接收message type，下一步就要接收message header**。

检查packet indicator：Command packet只能是从Host到Controller，Event packet只能是从Controller到Host，其余均可双向传输。（P1658）

如果message type是Command、ACL和SYNC其中一个则正常。

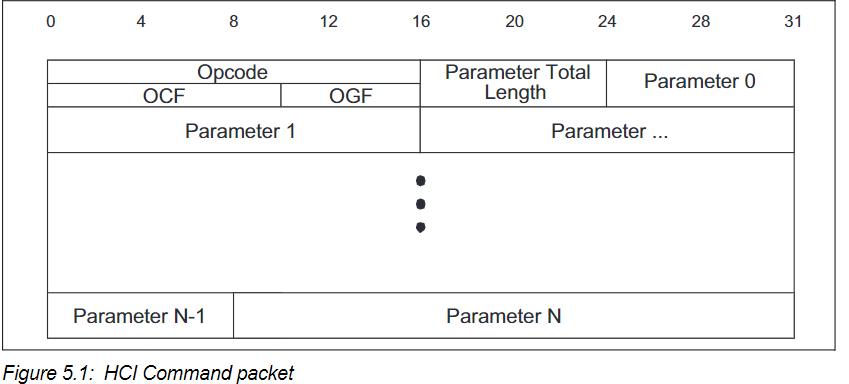


所以将当前HCI的RX state设置为HCI\_STATE\_RX\_HDR，等待接收后面的Header。

否则会设置state为HCI\_STATE\_RX\_OUT\_OF\_SYNC，此时同步错误，需要发送HCI\_RESET命令，之后开启下一轮的读取。

1. **HCI\_STATE\_RX\_HDR：获取到了Header message**。

以CMD为例：首先会进行提取Header各组成部分（是hci\_eif\_rx\_cmd\_hdr\_extract），并检查parlen，如果parlen为0，表示后面没有需要读取的参数了，此次数据接收完成，进行相应的处理即可。如果parlen >= 1，则需要进行后面paylod的读取。(P1810)

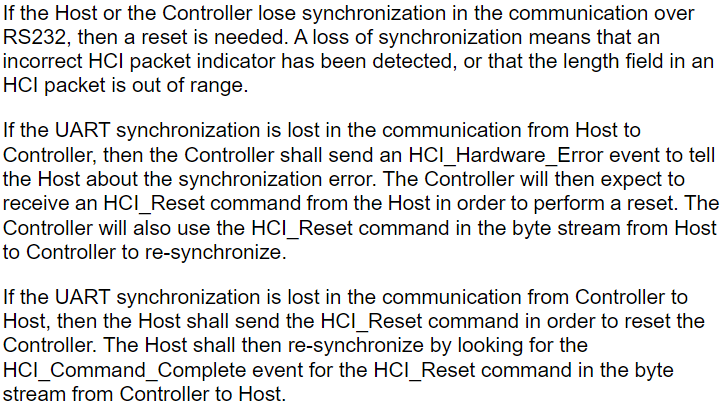


具体操作是首先判断parlen是否为一个超过最大限定值的数，没有超过时会在内核中申请相对应长度的空间，并将首地址赋值给hci\_env中的payl\_buff指针，之后进入第三种类型接收payload操作。

1. HCI\_STATE\_RX\_PAYL：接收Payload

根据接收到hci\_env中的各类参数，比如CMD header中的ogf、ocf、parlen，或者是ACL的header中的connection handle和flags等等，通过发送msg进行对应的处理。

1. HCI\_STATE\_OUT\_OF\_SYNC:异常处理(P1660)



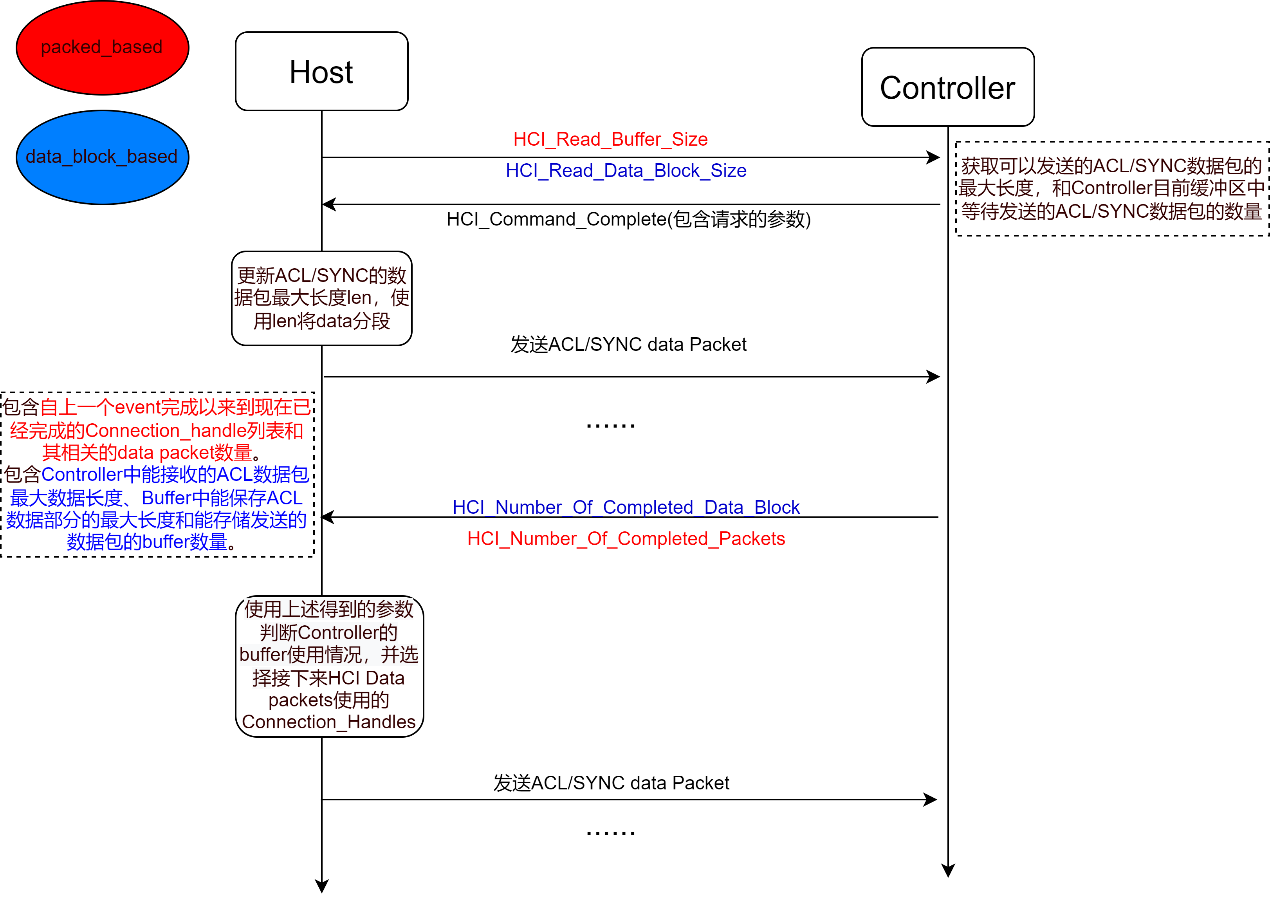
Controller会发出HCI\_Hardware\_Error事件给到Host端，Host通过发送HCI\_Reset命令给Controller来完成HCI的reset，之后会开启新一轮的数据读取流程。

## 流量控制

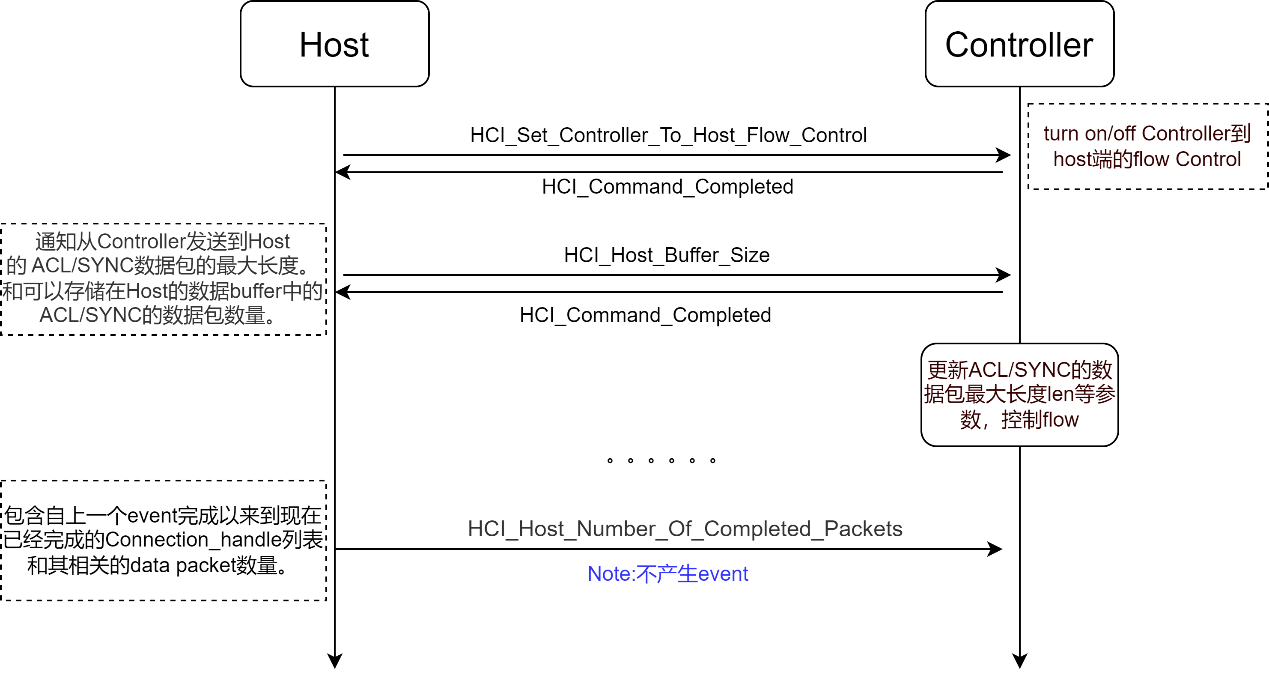
### 1）Host到Controller

HCI中Host到Controller的data flow Control有两种方式：“packet-based” 和 “data-block-based”。

在Host发送数据之前，Host会发送命令来查看controller能支持的数据包长度和缓冲区情况，这个命令可以是HCI\_Read\_Buffer\_Size或者HCI\_Read\_Data\_Block\_Size。在获得了数据包长度等参数之后，Host端可以向Controller端发送ACL/SYNC数据，同时当与另一个设备至少有一个连接时，或者在BR/EDR控制器上的本地环回模式下时，Controller可以使用HCI\_Number\_Of\_Completed\_Packets或者HCI\_Number\_Of\_Completed\_Data\_Block event来控制Host端的数据流量。



### 2）Controller到Host



Host使用HCI\_Host\_Number\_Of\_Completed\_Packets命令的参数等与Controller使用HCI\_Number\_Of\_Completed\_Packets事件基本相同，但是当主机准备释放相应的缓冲区时，一个数据包就完成了。

Note：**HCI\_Host\_Number\_Of\_Completed\_Packets**是一个特殊命令，不使用命令流控制，并且可以随时发送连接或在本地loopback模式下发送。命令完成后，该命令也不会产生事件。这使得流控制在两个方向上都可以完全相同，并且普通命令的流程不会受到干扰。