```
////////////////////////////////////////////////////////////////////////////////////////
//// This was ran on an STM32H743 Nucleo Board
//// STM32CubeMX V4.26.1
//// STM32CubeH7 V1.3.0
////////////////////////////////////////////////////////////////////////////////////////
/******** How to read the traces
  **** Always starts with HAL_ETH_IRQHandler() calling HAL_ETH_IsRxDataAvailable()
         // called from HAL_ETH_IRQHandler()
00018495>     HAL_ETH_IsRxDataAvailable() 1stRxDesc[9/10] Rx_Buff 30003798 // Rx_Buff[]
  **** Then ethernetif_input() is woken up via the semaphore
00018495> ethernetif_input() Start
  ****                        pbuf[index] address payload_address p->flags
  ****                           |          |           |           |
  ****                           |          |           |           | status (OK or dup)
  ****                           |          |           |           |  |  pbufs in-use
  ****                           |          |           |           |  |  |
  ****                           |          |           |           |  |  |
00018495> |   low_level_input() p[09] 2405C184     pl 30003798     02 OK  1
  ****                                                          expected payload
  ****                                                                |
00018495> |     low_level_input() p[09] 2405C184 pl 30003798 == 30003798  // correct
  **** After each packet ethernetif_input() rebuilds the processed descriptor
  ****  NOTE: The correct Rx_Buff[] is reassigned to the descriptor!
00018497> |    HAL_ETH_BuildRxDescriptors() CurRxDesc[9] 300000D8 Rx_Buff 30003798
          |
00018498> |    HAL_ETH_IsRxDataAvailable() No received packets
  **** When all frames are received, ethernetif_input() exits
00018498> ethernetif_input() Done
  **** Packets go up the stack and are freed by pbuf_free_custom()
00018498> pbuf_free_custom() p 2405C184 pl 300037BA 0A cnt 0
*****************************************************************************/
```

```
////////////////////////////////////////////////////////////////////////
////////////////////// On a busy network ///////////////////////////////
////////////////////////////////////////////////////////////////////////
///////// Most of the time a single packet is received, processed and freed.
----------------------- This is a single good frame -------------------------
          // called from HAL_ETH_IRQHandler()
00018495>     HAL_ETH_IsRxDataAvailable() 1stRxDesc[9/10] Rx_Buff 30003798 // Rx_Buff[]
00018495> ethernetif_input() Start
00018495> |   low_level_input() p[09] 2405C184 pl 30003798 02 OK  1
00018495> |     low_level_input() p[09] 2405C184 pl 30003798 == 30003798 // correct Rx_Buff
00018497> |    HAL_ETH_BuildRxDescriptors() CurRxDesc[9] 300000D8 Rx_Buff 30003798
          |
00018498> |    HAL_ETH_IsRxDataAvailable() No received packets
00018498> ethernetif_input() Done
00018498> pbuf_free_custom() p 2405C184 pl 300037BA 0A cnt 0
--------------------------- all cleaned up --------------------------------

---------- Sometimes multiple packets (2) are received and this happens --------
          // called from HAL_ETH_IRQHandler()
00018670>     HAL_ETH_IsRxDataAvailable() 1stRxDesc[10/11] Rx_Buff 30003D90 // Rx_Buff[]
00018671> ethernetif_input() Start
00018671> |   low_level_input() p[10] 2405C198 pl 30003D90 02 OK  1
00018671> |     low_level_input() p[10] 2405C198 pl 30003D90 == 30003D90 // correct Rx_Buff
00018673> |    HAL_ETH_BuildRxDescriptors() CurRxDesc[10] 300000F0 Rx_Buff 30003D90
          |
00018673> |    HAL_ETH_IsRxDataAvailable() 1stRxDesc[11/12] Rx_Buff 30004388 // Rx_Buff[]
00018673> |  *low_level_input() p[11] 2405C1AC pl 30003D90 02 dup 2 // WRONG Rx_Buff
00018673> |     low_level_input() p[11] 2405C1AC pl 30003D90 != 30004388 // using previous
00018675> |    HAL_ETH_BuildRxDescriptors() CurRxDesc[11] 30000108 Rx_Buff 30004388 // fixed
          |
00018676> |    HAL_ETH_IsRxDataAvailable() No received packets
00018676> ethernetif_input() Done
00018676> pbuf_free_custom() p 2405C198 pl 30003D9E 0A cnt 1
00018678> pbuf_free_custom() p 2405C1AC pl 30003D9E 0A cnt 0
--------------------------- all cleaned up --------------------------------
```

```
----------------------- I sent 10 back-to-back UDP Datagrams ----------------
**** The extra line is the UDP data which starts with a 2-byte binary sequence (03 D4) then text
00020239> |    low_level_input: p[03] 2405C10C 300013F2 03 D4 33 34 35 36 37 38 39 30

             // called from HAL_ETH_IRQHandler()
00020239>     HAL_ETH_IsRxDataAvailable() 1stRxDesc[3/4] Rx_Buff 300013C8 // Rx_Buff[]
00020239> ethernetif_input() Start
00020239> |   low_level_input() p[03] 2405C10C pl 300013C8 02 OK  1
00020239> |     low_level_input() p[03] 2405C10C pl 300013C8 == 300013C8 // correct Rx_Buff
00020239> |    low_level_input: p[03] 2405C10C 300013F2 03 D4 33 34 35 36 37 38 39 30
00020241> |    HAL_ETH_BuildRxDescriptors() CurRxDesc[3] 30000048 Rx_Buff 300013C8
          |
00020241> |    HAL_ETH_IsRxDataAvailable() 1stRxDesc[4/5] Rx_Buff 300019C0 // Rx_Buff[]
00020242> |  *low_level_input() p[04] 2405C120 pl 300013C8 02 dup 2
00020242> |     low_level_input() p[04] 2405C120 pl 300013C8 != 300019C0 // using previous
00020242> |    low_level_input: p[04] 2405C120 300013F2 03 D4 33 34 35 36 37 38 39 30
00020244> |    HAL_ETH_BuildRxDescriptors() CurRxDesc[4] 30000060 Rx_Buff 300019C0 // fixed
          |
00020244> |    HAL_ETH_IsRxDataAvailable() 1stRxDesc[5/6] Rx_Buff 30001FB8 // Rx_Buff[]
00020244> |   low_level_input() p[05] 2405C134 pl 300019C0 02 OK  3
00020244> |     low_level_input() p[05] 2405C134 pl 300019C0 != 30001FB8 // using previous
00020244> |    low_level_input: p[05] 2405C134 300019EA 03 D5 33 34 35 36 37 38 39 30
00020247> |    HAL_ETH_BuildRxDescriptors() CurRxDesc[5] 30000078 Rx_Buff 30001FB8 // fixed
          |
00020247> |    HAL_ETH_IsRxDataAvailable() 1stRxDesc[6/7] Rx_Buff 300025B0 // Rx_Buff[]
00020247> |   low_level_input() p[06] 2405C148 pl 30001FB8 02 OK  4
00020247> |     low_level_input() p[06] 2405C148 pl 30001FB8 != 300025B0 // using previous
00020247> |    low_level_input: p[06] 2405C148 30001FE2 03 D6 33 34 35 36 37 38 39 30
00020249> |    HAL_ETH_BuildRxDescriptors() CurRxDesc[6] 30000090 Rx_Buff 300025B0 // fixed
          |
00020249> |    HAL_ETH_IsRxDataAvailable() 1stRxDesc[7/8] Rx_Buff 30002BA8 // Rx_Buff[]
00020249> |   low_level_input() p[07] 2405C15C pl 300025B0 02 OK  5
00020250> |     low_level_input() p[07] 2405C15C pl 300025B0 != 30002BA8 // using previous
00020250> |    low_level_input: p[07] 2405C15C 300025DA 03 D7 33 34 35 36 37 38 39 30
00020252> |    HAL_ETH_BuildRxDescriptors() CurRxDesc[7] 300000A8 Rx_Buff 30002BA8 // fixed
          |
00020252> |    HAL_ETH_IsRxDataAvailable() 1stRxDesc[8/9] Rx_Buff 300031A0 // Rx_Buff[]
00020252> |   low_level_input() p[08] 2405C170 pl 30002BA8 02 OK  6
00020252> |     low_level_input() p[08] 2405C170 pl 30002BA8 != 300031A0 // using previous
```

```
00020252> |    low_level_input: p[08] 2405C170 30002BD2 03 D8 33 34 35 36 37 38 39 30
00020255> |    HAL_ETH_BuildRxDescriptors() CurRxDesc[8] 300000C0 Rx_Buff 300031A0 // fixed
          |
00020255> |    HAL_ETH_IsRxDataAvailable() 1stRxDesc[9/10] Rx_Buff 30003798 // Rx_Buff[]
00020255> |   low_level_input() p[09] 2405C184 pl 300031A0 02 OK  7
00020255> |     low_level_input() p[09] 2405C184 pl 300031A0 != 30003798 // using previous
00020255> |    low_level_input: p[09] 2405C184 300031CA 03 D9 33 34 35 36 37 38 39 30
00020257> |    HAL_ETH_BuildRxDescriptors() CurRxDesc[9] 300000D8 Rx_Buff 30003798 // fixed
          |
00020257> |    HAL_ETH_IsRxDataAvailable() 1stRxDesc[10/11] Rx_Buff 30003D90 // Rx_Buff[]
00020257> |   low_level_input() p[10] 2405C198 pl 30003798 02 OK  8
00020258> |     low_level_input() p[10] 2405C198 pl 30003798 != 30003D90 // using previous
00020258> |    low_level_input: p[10] 2405C198 300037C2 03 DA 33 34 35 36 37 38 39 30
00020260> |    HAL_ETH_BuildRxDescriptors() CurRxDesc[10] 300000F0 Rx_Buff 30003D90 // fixed
          |
00020260> |    HAL_ETH_IsRxDataAvailable() 1stRxDesc[11/12] Rx_Buff 30004388 // Rx_Buff[]
00020260> |   low_level_input() p[11] 2405C1AC pl 30003D90 02 OK  9
00020260> |     low_level_input() p[11] 2405C1AC pl 30003D90 != 30004388 // using previous
00020260> |    low_level_input: p[11] 2405C1AC 30003DBA 03 DB 33 34 35 36 37 38 39 30
00020262> |    HAL_ETH_BuildRxDescriptors() CurRxDesc[11] 30000108 Rx_Buff 30004388 // fixed
          |
00020263> |    HAL_ETH_IsRxDataAvailable() 1stRxDesc[12/13] Rx_Buff 30004980 // Rx_Buff[]
00020263> |   low_level_input() p[12] 2405C1C0 pl 30004388 02 OK  10
00020263> |     low_level_input() p[12] 2405C1C0 pl 30004388 != 30004980 // using previous
00020263> |    low_level_input: p[12] 2405C1C0 300043B2 03 DC 33 34 35 36 37 38 39 30
..34567890
00020265> |    HAL_ETH_BuildRxDescriptors() CurRxDesc[12] 30000120 Rx_Buff 30004980 // fixed
          |
00020265> |    HAL_ETH_IsRxDataAvailable() No received packets
          |   // 10 UDPs were sent. Because the 1st was duplicated into the 2nd,
          |   //  the 10th was lost.
00020265> ethernetif_input() Done
00020265> pbuf_free_custom() p 2405C10C pl 300013D6 02 cnt 9
00020268> pbuf_free_custom() p 2405C120 pl 300013D6 02 cnt 8  // duplicate payload
00020270> pbuf_free_custom() p 2405C134 pl 300019CE 02 cnt 7
00020272> pbuf_free_custom() p 2405C148 pl 30001FC6 02 cnt 6
00020274> pbuf_free_custom() p 2405C15C pl 300025BE 02 cnt 5
00020277> pbuf_free_custom() p 2405C170 pl 30002BB6 02 cnt 4
00020279> pbuf_free_custom() p 2405C184 pl 300031AE 02 cnt 3
```

00020281> pbuf_free_custom() p 2405C198 pl 300037A6 02 cnt 2

00020283> pbuf_free_custom() p 2405C1AC pl 30003D9E 02 cnt 1

00020286> pbuf_free_custom() p 2405C1C0 pl 30004396 02 cnt 0

//// Notice that after the duplicate payload, each payload is offset by +1 Rx_Buff[].

---------- all cleaned up ----------