Frame and Packet definitions throughout the design

Throughout the design, the words “frame” and “packet” are used extensively. However, on different levels (in the OSI model) they mean different things and it is quite easy to get confused. Therefore a listing of their meaning is given below.
Figure 1 shows the definitions of “frame” and “packet” on the physical layer of the OSI-model as defined in IEEE802.3-2008.

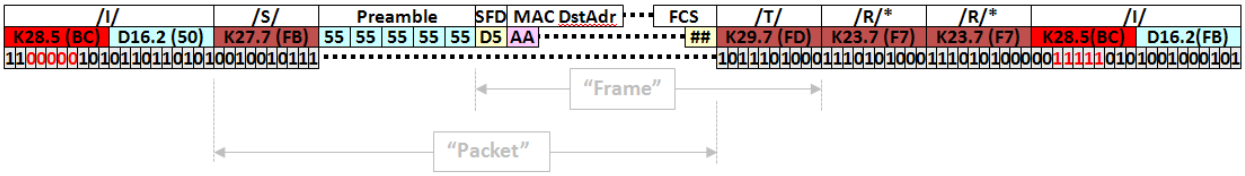

Figure 1: Definitions of “frame” and “packet” on the *physical* layer of the OSI-model as defined in IEEE802.3-2008 Clause 30 (see also 46.2.5)

On the *data link* layer, where the transactions with the MAC take place, a “frame” is defined as all data that are signaled “rx_data_valid” by the MAC. These data range from the first byte after the Start of Frame Delimiter (SFD) to the last data byte before the first Frame Control Sequence (FCS); see figure 2.

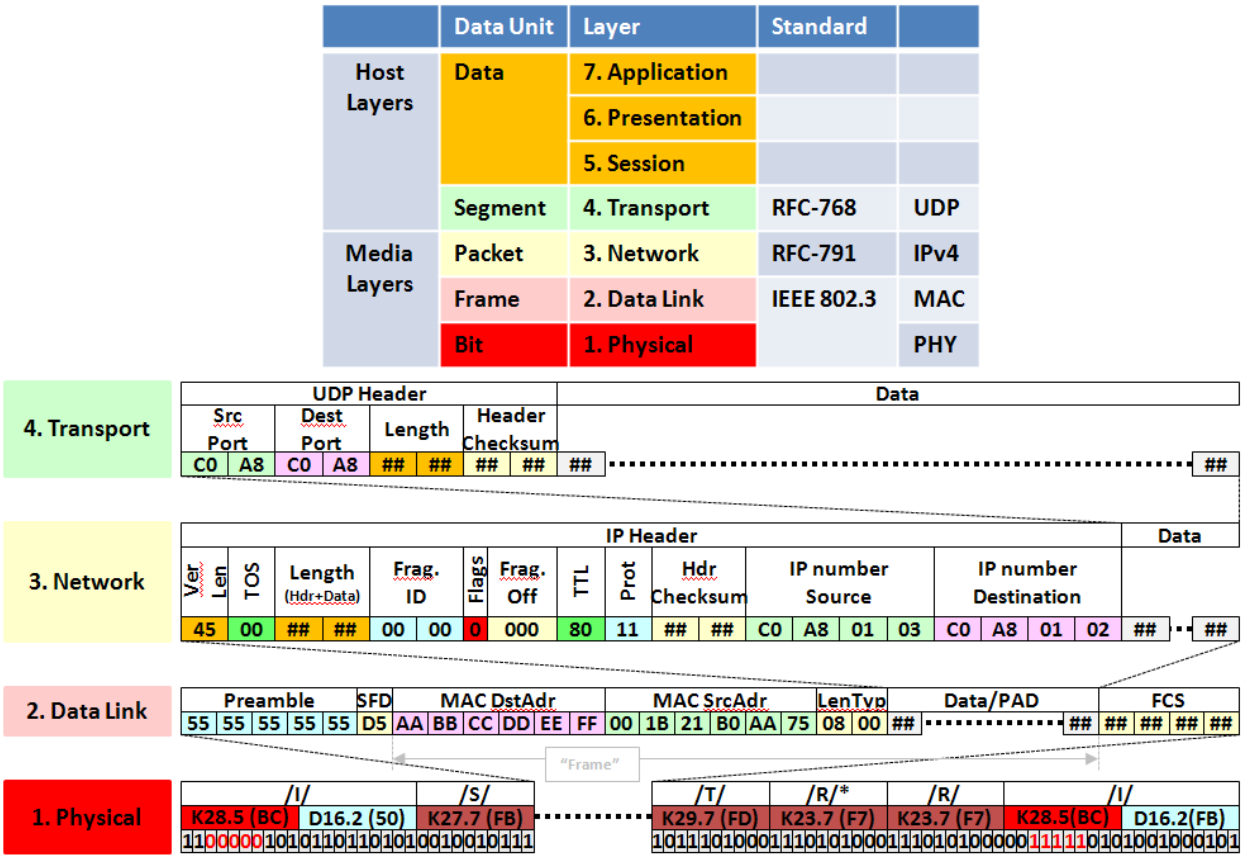

Figure 2: “frame” and “packet” throughout the levels of the OSI-Model

Note that for Ethernet v2 framing (see also: http://en.wikipedia.org/wiki/EtherType) this also includes padded data that might have been inserted to ensure that the frame length is at least 64 bytes.

On the *network* layer, the term “packet” is usually associated with IP packets that consist of an IP header and IP data.
At the level of the transport layer, the term “packet” is usually associated with an IP packet (i.e. IP header plus IP data) where the IP data is formatted according to the TCP or UDP protocol for example.

At the level of data transactions to and from the medipix chips the “EnableIn” signal initiated a “frame” and the “EnableOut” signal terminates a “frame”

Tx Packet Generator

Either the Tx packet generator is waiting for data from the input FIFO or it is waiting for an access request by the CPU. A CPU access has priority.

In case the input FIFO has data then:

1. If there is space in the Tx buffer for at least one jumbo frame (9014 = 9000 bytes + 14 MAC header bytes)
2. If there is no CPU Access Request is pending
3. If there is data to transmit

then data will be transferred from the input FIFO to the Tx Buffer leaving some space to place the MAC header, IP header and UDP-header later *in front of* the data. While the data is transferred the UDP Checksum is calculated.
If all data is transferred (an EOF is encountered) then the UDP header is written in front of the data. This includes Source port, Destination port, UDP-Length and UDP Checksum.
Next the IP header is stored in front of the UDP header. This includes IP Length and header checksum.
Finally the MAC header is stored in front of the IP header. MAC Source and Destination address are input to the state machine and originate from constants or registers.
The packet pointer are updated and the empty flag signals that a packet is available for transmission via the MAC.

The CPU may be granted access in order to be able to create arbitrary packets, serving arbitrary protocols.
If there is a CPU access request then the CPU is granted access as soon as:

1. The current assembly of a packet in the Tx buffer has finished
2. There is space in the Tx buffer for at least one jumbo frame (9014 = 9000 bytes + 14 MAC header bytes)

After being granted access, the CPU writes data (16-bit words). The CPU address serves as an offset into the Tx packet that is being assembled by the CPU. The physical Tx buffer address is automatically calculated based on the Tx buffer pointers plus the CPU address. The last word of the packet must be accompanied with an End Of Packet (EOP) (note, since the CPU has random access to the packet buffer this doesn’t necessarily mean that the last word of the packet is actually written last to the packet buffer).
The CPU is responsible for generating the proper packet data format (i.e. MAC, IP etc. headers, checksums and data).

Rx Packet checks

As soon as a packet is received and the Rx buffer signals “not-empty” then it is assumed that a MAC, IPv4 , UDP type header is received (in total 42 bytes). This header is read from the Rx buffer and several properties (see

Table 1) are checked. If one of the checks fail than a bit in the Error Vector is set and the content of the buffer is passed to the CPU for further investigation.

Error Vector bit	Check
0	Ether Type = IPv4 (x0800)
1	IP header checksum
2	IP Version = 4 and IP Header length = 5
3	Protocol = UDP (x11)

Table 1: The Error Vector bit is set when the check fails

If the packet passes all the checks and the UDP destination port matches one of the predefined ports (originating from either constants or registers) then only the payload of the packet is routed to the corresponding output stream. All other packets (all headers etc. included) are routed to the CPU.