Low Level Data Transfer Protocol

The low level data transfer protocol has the following properties, which makes it suitable for use with a range of physical implementations and as a bearer for a range of higher level protocols:

1. Byte Oriented Protocol : All transfers consist of an integer number of consecutive bytes, making the protocol suitable for on-device communications as well as via fast byte oriented inter-chip serial communication links.
2. Little Endian Format : Multi-byte data items can be included at any point in the sequence of bytes using little endian formatting (least significant byte first).
3. SELF Based Flit Flow Control : By default, flow control occurs at the flit level using the SELF handshake mechanism. More sophisticated flow control can be incorporated for buffered links, provided that the flit based flow control mechanism is preserved at the boundaries.
4. Out-Of-Band Framing : By default, frame delineation is implemented out-of-band from the transmitted data stream in order to minimise the amount of in-band processing during transit. In-band framing may be incorporated using HDLC style frame delimiters and octet escaping for transport over external serial links or while the data is at rest in memory.
5. Scalable Flit Widths : In the basic flit format, each flit is assumed to contain a single byte. However, for increased throughput bytes may be packed into flits that are 2, 4, 8, 16, 32, 64 or 128 bytes wide with the convention that the first byte is packed into byte position zero (preserving little endian ordering).

Basic Flit Based Transfers

For basic byte wide transfers, each flit consists of an eight bit data signal (DATA) and a single bit end of frame control signal (EOF). The EOF signal is asserted in conjunction with the final DATA byte of a frame to provide frame delineation.

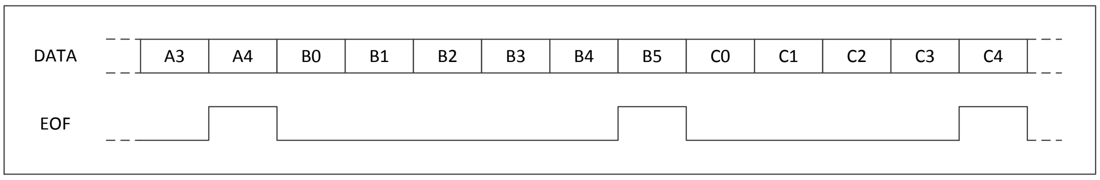

Figure 1 : Basic Single Byte Flit Transfers

Wide Flit Based Transfers

For wide flit based transfers, each flit consists of a data signal that may be 2, 4, 8, 16, 32, 64 or 128 bytes wide (DATA2, DATA4, DATA8, DATA16, DATA32, DATA64 or DATA128) and a single byte end of frame control signal (EOFC). The EOFC signal is normally set to zero, but is set to a non-zero value in conjunction with the final DATA*X* word of a frame to provide frame delineation. When used to indicate the end of a frame, the non-zero value of the EOFC signal indicates the number of bytes within the corresponding DATA*X* word which contain valid data. Any bytes which do not contain valid data may have arbitrary values. Note that while the EOFC signal is always a nominal eight bits wide, for data signal widths less than 128 bytes wide the upper bits of the EOFC signal will always be set to zero – allowing hardware implementations to be optimised accordingly.

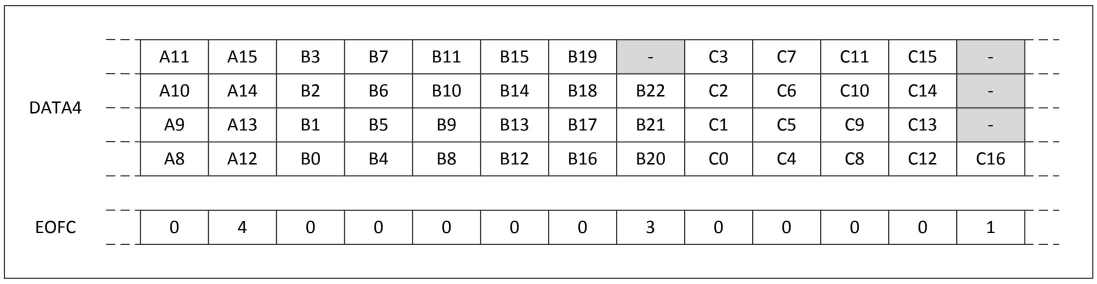
DATA4
A11 A15 B3 B7 B11 B15 B19 - C3 C7 C11 C15 -
A10 A14 B2 B6 B10 B14 B18 B22 C2 C6 C10 C14 -
A9 A13 B1 B5 B9 B13 B17 B21 C1 C5 C9 C13 -
A8 A12 B0 B4 B8 B12 B16 B20 C0 C4 C8 C12 C16
EOFC
0 4 0 0 0 0 0 3 0 0 0 0 1

Figure 2 : Wide Multi-Byte Flit Transfers

Simple In-Band Framing

The simple in-band framing support is provided for situations where it is not possible to support the out-of-band control signals and where there is not an existing protocol for in-band signalling that can be used. An example would be where data frames are being transferred using a circular buffer in shared memory. The standard HDLC frame delimiter of 0x7E is used to indicate the start and end of a frame, as well as being used as an idle value if required. Standard HDLC octet escaping is used in situations where the frame delimiter occurs in the transmitted data, such that every occurrence of the octet 0x7E is replaced by the two octets 0x7D and 0x5E. The HDLC header and frame check sequence bytes are not used in this context.

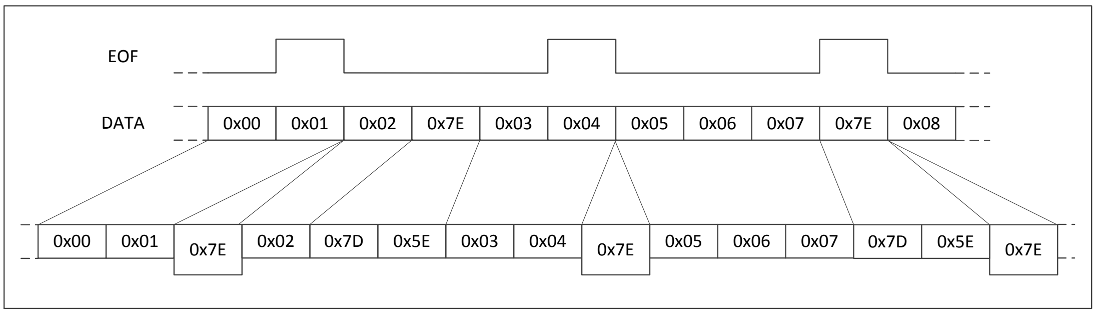
EOF
DATA
0x00 0x01 0x02 0x7E 0x03 0x04 0x05 0x06 0x07 0x7E 0x08
0x00 0x01 0x7E 0x02 0x7D 0x5E 0x03 0x04 0x7E 0x05 0x06 0x07 0x7D 0x5E 0x7E

Figure 3 : Mapping To HDLC Style In-Band Frame Delimiters

Flit Width Adaptation

The use of out-of-band framing delimiters allows the flit size of a given data link to be easily scaled, which provides a corresponding scaling of the bandwidth that can be accommodated. This is particularly useful for applications such as arbitrated memory accesses, where large numbers of lower bandwidth connections are arbitrated onto a single high bandwidth memory bus.

When adapting to a larger flit width, the source data is striped across the larger flit data bus in the manner shown in Figure 2. At the end of the frame, word alignment is then enforced by adding the required number of padding bytes, which may have arbitrary values.

When adapting to a smaller flit width, the source data is split out from the larger flit data bus and forwarded on the narrower bus with the least significant byte position(s) first. Any narrow flits generated in this manner which consist entirely of padding bytes should be discarded, so that the last narrow flit in the frame has at least one data byte present.

Message Type Identifiers

The first byte of each frame will be a message type identifier that can be used to determine the formatting of the remaining frame contents. By convention, whenever a pair of message types form a request/response pair, the message type identifier for the response will be the ones complement of the corresponding request identifier.

TypeID	Direction	Description
0x00	None	Reserved
0xFF	None	Reserved
0x01	Request	Memory write request
0xFE	Response	Memory write response
0x02	Request	Memory read request
0xFD	Response	Memory read response

Table 1 : Message Type Identifiers

Memory Access Protocol

The memory access protocol is implemented on top of the low level transfer protocol in order to support arbitrated read and write access to blocks of shared RAM. All memory access transactions have the following properties:

1. All transfers are treated as address incrementing bursts consisting of an integer number of bytes.
2. Individual burst transfers must not cross 4KB address boundaries in order to support interoperability with AXI based memory infrastructure.
3. Memory accesses are not assumed to be uniform, since the same bus infrastructure may be used for accessing memory areas that use different RAM technologies. This will require protocol support for out of order delivery of response messages.
4. Write responses are used to indicate that a write operation has committed the data to memory in such a way that subsequent memory consistency is guaranteed. This may include committing the data to an intermediate data cache layer that supports full cache coherence.

Memory Request Message Format

The memory access protocol uses a common message format for both read and write requests, with the only difference being whether the appended write data field contains valid data. The message format used is shown in the following table.

Field	Offset	Size	Description
Message Type	0x00	1	Message type (see Table 1).
Options	0x01	1	Request options.
Message Tag	0x02	2	Message tag for request/response matching.
Base Address	0x04	8	Base address in memory for data transfer.
Burst Length	0x0C	2	Data transfer burst length (N).
Write Data	0x0E	0/N	Write data (write requests only).

Table 2 : Memory Access Request Format

Memory Response Message Format

The memory access protocol uses a common message format for both read and write responses, with the only difference being whether the appended read data field contains valid data. The message format used is shown in the following table.

Field	Offset	Size	Description
Message Type	0x00	1	Message type (See Table 1)
Status	0x01	1	Response status.
Message Tag	0x02	2	Message tag for request/response matching.
Read Data	0x04	0/N	Read data (read responses only).

Table 3 : Memory Access Response Format