LocalLink Interface Specification

SP006 (v2.0) July 25, 2005

XILINX®

Xilinx is disclosing this Specification (hereinafter “the Specification”) to you for use in the development of designs to operate on, or interface with, Xilinx FPGAs.

Xilinx does not assume any liability arising out of the application or use of the Specification; nor does Xilinx convey any license under its patents, copyrights, or any rights of others. You are responsible for obtaining any rights you may require for your use or implementation of the Specification. Xilinx reserves the right to make changes, at any time, to the Specification as deemed desirable in the sole discretion of Xilinx. Xilinx assumes no obligation to correct any errors contained herein or to advise you of any correction if such be made. Xilinx will not assume any liability for the accuracy or correctness of any engineering or technical support or assistance provided to you in connection with the Specification.

THE SPECIFICATION IS PROVIDED “AS IS" WITH ALL FAULTS, AND THE ENTIRE RISK AS TO ITS FUNCTION AND IMPLEMENTATION IS WITH YOU. YOU ACKNOWLEDGE AND AGREE THAT YOU HAVE NOT RELIED ON ANY ORAL OR WRITTEN INFORMATION OR ADVICE, WHETHER GIVEN BY XILINX, OR ITS AGENTS OR EMPLOYEES. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE SPECIFICATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOST DATA AND LOST PROFITS, ARISING FROM OR RELATING TO YOUR USE OF THE SPECIFICATION, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE TOTAL CUMULATIVE LIABILITY OF XILINX IN CONNECTION WITH YOUR USE OF THE SPECIFICATION, WHETHER IN CONTRACT OR TORT OR OTHERWISE, WILL IN NO EVENT EXCEED THE AMOUNT OF FEES PAID BY YOU TO XILINX HEREUNDER FOR USE OF THE SPECIFICATION. YOU ACKNOWLEDGE THAT THE FEES, IF ANY, REFLECT THE ALLOCATION OF RISK SET FORTH IN THIS AGREEMENT AND THAT XILINX WOULD NOT MAKE AVAILABLE THE SPECIFICATION TO YOU WITHOUT THESE LIMITATIONS OF LIABILITY.

The Specification is not designed or intended for use in the development of on-line control equipment in hazardous environments requiring fail-safe controls, such as in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, life support, or weapons systems (“High-Risk Applications”). Xilinx specifically disclaims any express or implied warranties of fitness for such High-Risk Applications. You hereby acknowledge that use of the Specification in such High-Risk Applications is fully at your risk.

LocalLink Interface Specification
SP006 (v2.0) July 25, 2005

The following table shows the revision history for this document.

	Version	Revision
10/18/02	1.0	Initial Xilinx release. This specification was originally released as DS230.
02/26/04	1.1	This revisions include: • Updated copyright information; made editorial changes • Changed document number to DS290
07/25/05	2.0	The document was converted from a data sheet (DS290) to a specification (SP006) This revision includes: • Updated legal page • Expanded rem functionality • Updated channelized frame section • New specification format • General update to more completely convey the intent of the specification

Table of Contents

3 Timing Diagrams

About This Specification

This document is a specification for the LocalLink interface, which is a system-level user interface to Xilinx intellectual property (IP) cores.

Document Contents

This document contains the following sections:

- Section 1, Overview, covers the features of the LocalLink interface and briefly describes the source and destination interfaces.
- Section 2, Detailed Interface Specification, provides greater detail about the interface signals.
- Section 3, Timing Diagrams, shows timing diagrams for various examples of data transfers.

1 Overview

1.1 Introduction

The LocalLink interface specification defines a high-performance, synchronous, point-to-point connection designed to serve as a user interface to system-level Xilinx intellectual property (IP) designs. The specification defines a set of signals that allows the transfer of frames and enables the functions listed in Section 1.2 Features. This specification is intended to support frames from any type of protocol.

1.2 Features

- User interface of arbitrary byte width
- Interface for framed data of arbitrary length
- Synchronous point-to-point communication
- Upstream and downstream flow control
- Channelization support
- Efficient link bandwidth utilization
- Parity checking and error reporting

1.3 Functional Description

Figure 1-1 is a high-level block diagram of the LocalLink interface and illustrates its basic topology. A LocalLink interface consists of a source and a destination interface. The source application communicates with the destination application in simplex mode. An application could be an IP core, a customer's application, or any design entity. For full duplex operation between two applications, two LocalLink interfaces, each consisting of a source and destination, need to be implemented. Table 1-1, page 11 provides a complete list of signals for the LocalLink interface.

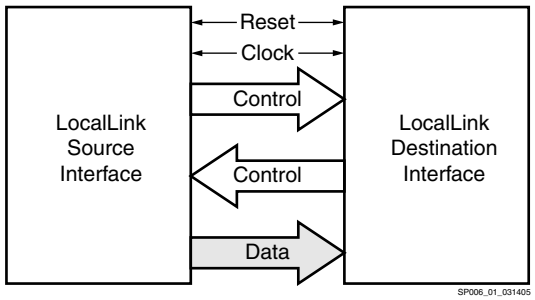

Figure 1-1: **LocalLink Interface Block Diagram**

1.4 Technical Conventions and Definitions

1.4.1 Vector Nomenclature

- **Bits in a vector:**
  - The leftmost bit in a vector is the most-significant bit (MSb)
  - The rightmost bit in a vector is the least-significant bit (LSb)
- **Bytes in a vector:**
  - The leftmost byte in a vector is the most-significant byte (MSB)
  - The rightmost byte in a vector is the least-significant byte (LSB)
- **Ordering of vector signals:**
  - The LocalLink interface supports two orderings of vector signals: *ascending* and *descending*. If *n* is the number of bits in the vector, then:
    - An ascending vector has 0 as the most-significant (leftmost) bit; for example *vector_name*[0:*n*-1]
    - A descending vector has *n*-1 as the most-significant (leftmost) bit; for example, *vector_name*[*n*-1:0]

1.4.2 Frame Elements

A LocalLink frame has up to three elements:

- **Payload:** User frame data
- **Header (optional):** User-defined; precedes payload
- **Footer (optional):** User-defined; follows payload

1.4.3 Bus Cycle Types

A LocalLink interface has two types of bus cycles:

- **Data beat:** A bus cycle that transfers data and has no framing control signals asserted
- **Control beat:** A bus cycle in which framing control signals are asserted. Data is usually transferred during control beats

1.4.4 Control Signaling

Control signals can be implemented as active-Low or active-High. This specification generally uses the active-Low convention, which is indicated by a trailing _N. A LocalLink frame has two types of control signals:

- **Framing control signals:** Mark the beginning and end of frame elements. There are four framing control signals: (1) A signal that marks the beginning of a frame, (2) the end of a frame, and optionally, (3) the beginning, and (4) the end of payload bytes.
- **Flow control signals:** Used by the source and destination to signal readiness to transact a data or control beat.

1.4.5 Numerical Conventions

- Hexadecimal numbers are indicated by a trailing, lowercase h

1.5 Pinout

The required signal names and descriptions are shown in Table 1-1 and the optional signal names and descriptions are shown in Table 1-2, page 12.

Table 1-1: **Required Interface Signals**

Signal Name	Direction	Description
CLK	Input	**Clock:** All signals are synchronous to this clock.
RST_N	Input	**Reset:** When reset is asserted, any frames in progress are aborted.
DATA(1)	src to dst	**Data Bus:** The data bus is driven by the source. Frame data is transmitted across this bus.
SRC_RDY_N	src to dst	**Source Ready:** Indicates that the source is ready to transact a data or control beat.
DST_RDY_N	dst to src	**Destination Ready:** Indicates that the destination is ready to transact a data or control beat.
SOF_N	src to dst	**Start-of-Frame:** Indicates a beat that contains the first byte(s) of a frame.
EOF_N	src to dst	**End-of-Frame:** Indicates a beat that contains the last byte(s) of a frame.

Notes:

1. The LocalLink interface supports two orderings of vector signals:
   - Ascending: Where 0 is the most-significant (leftmost) bit; for example *vector_name*[0:*n*–1]
   - Descending: Where *n*–1 is the most-significant (leftmost) bit; for example, *vector_name*[*n*–1:0]

   (*n* is the number of bits in the vector)

The optional signal names and descriptions are shown in Table 1-2.

Table 1-2: **Optional Interface Signals**

Signal Name	Direction	Description
SOP_N	src to dst	**Start-of-Payload:** Indicates a beat that contains the first payload byte(s) of a frame, if the frame has a header.
EOP_N	src to dst	**End-of-Payload:** Indicates a beat that contains the last payload byte(s) of a frame, if the frame has a footer.
REM(1)	src to dst	**End-of-Frame Remainder Bus:** Used primarily to indicate the position of the last byte of a frame in a word that is accompanied by an asserted EOF_N. If in a particular implementation, the beginning and end of all frame elements are always *unambiguous* without the aid of a remainder bus, then no remainder bus is needed. If the beginning and end of all frame elements are *ambiguous*, then one or more remainder buses must be used.
REM_SOF(1)	src to dst	**Start-of-Frame Remainder Bus:** Indicates the position of the first byte of a frame in a word that is accompanied by an asserted SOF_N.
REM_SOP(1)	src to dst	**Start-of-Payload Remainder Bus:** Indicates the position of the first payload byte in a word that is accompanied by an asserted SOP_N.
REM_EOP(1)	src to dst	**End-of-Payload Remainder Bus:** Indicates the position of the last payload byte in a word that is accompanied by an asserted EOP_N.
SRC_DSC_N	src to dst	**Source Discontinue:** Indicates that the source device is cancelling this frame due to errors or other causes.
DST_DSC_N	dst to src	**Destination Discontinue:** Indicates that the destination device is refusing to accept this frame due to insufficient buffer space or other causes.
CH(1)	These signals are driven by either the source or destination interface, depending upon the application. See Table 2-26, page 28	**Channel Number Bus:** Used for implementations that allow read and/or write access to a specific channel. This binary encoded vector indicates which channel a data transfer request or response is coming from or going to.
CH_RDY_N(1)		**Channel Ready Bus:** This vector indicates frame availability on the channels.
CH_SWAP_LOOKAHEAD_N(1)		**Channel Swap Lookahead Bus:** When asserted, the driving interface indicates that the current channel will go into an underflow state *soon*. (*Soon* is defined by the CH_LOOKAHEAD_CLK_CNT parameter described in Section 2.8.3 Channel Swap Lookahead, page 29.)
PARITY(1)	src to dst	**Parity Signal Bus:** Indicates the parity that is calculated over entire data bus on a byte basis.
PARITY_ERR_N	dst to src	**Parity Error:** Indicates that a parity error was detected during the previous data beat.

Notes:

1. The LocalLink interface supports two orderings of vector signals:
   - Ascending: Where 0 is the most-significant (leftmost) bit; for example *vector_name*[0:*n*–1]
   - Descending: Where *n*–1 is the most-significant (leftmost) bit; for example, *vector_name*[*n*–1:0]

   (*n* is the number of bits in the vector)

2 Detailed Interface Specification

2.1 Clock

Table 2-1 shows the definition and direction of the clock signal.

Table 2-1: **Clock Signal**

Signal Name	Direction	Description
CLK	Input	**Clock:** All signals are synchronous to this clock. All data bus and control signals must meet setup and hold times with respect to: • Rising edge of this clock if single data rate flip-flops are used • Both edges of this clock if double data rate flip-flops are used

2.2 Reset

Table 2-2 shows the definition and direction of the reset signal.

Table 2-2: **Reset Signal**

Signal Name	Direction	Description
RST_N	Input	**Reset:** When reset is asserted, any frames in progress are aborted. The source and destination have the option of resetting their internal logic to an initial state.

2.3 Data Bus

The data bus is used to transfer data from a LocalLink source to a LocalLink destination. Its width is always a multiple of 8 bits. Table 2-3 shows the direction and definition of the data bus vector.

Table 2-3: **Data Bus**

Signal Name	Direction	Description
DATA[*d*–1:0] or DATA[0:*d*–1]	src to dst	**Data Bus:** The data bus is driven by the source. Frame data is transmitted across this bus. The variable *d* is the number of data bits and is a multiple of 8.

The data bus places bytes in frame order. The first byte in a beat occupies the highest order bits. In a *d*-bit bus, either DATA[0:7] or DATA[*d*-1:*d*-8]. See Table 2-4 for data ordering detail.

Table 2-4: **Data Ordering in a *d*-Bit Bus**

Bit Ordering		MSB	. . .		LSB
Ascending	MSb is 0	0:7	8:15	. . .	*d*–8:*d*–1
Descending	MSb is *d*–1	*d*–1:*d*-8	. . .	15:8	7:0

2.4 Source and Destination Ready

2.4.1 Overview

The LocalLink interface allows the data source and data destination interfaces to regulate data flow with a simple handshake protocol. Although the source and destination can monitor the bus state at any time, only when they both assert their ready signals is a *data beat* or a *control beat* transacted.

The data source asserts its SRC_RDY_N signal to indicate its readiness to transact a bus cycle. The data destination asserts its DST_RDY_N signal to indicate its readiness to transact a bus cycle. For any bus cycle to occur, both source and destination ready signals must be asserted; the order of assertion does not matter.

Note: The reset signal has precedence over all other signals, including the clock. Reset causes the bus to immediately reinitialize.

Typically, the data source signifies that the control signals and the data bus are ready and should be acted upon by asserting SRC_RDY_N. The source can assert SRC_RDY_N first because it does not need to wait for its partner's ready signal to be asserted before asserting its own ready signal. The recipient of the bus signals (destination) indicates it is ready to act on LocalLink control and data by asserting DST_RDY_N. The destination can assert DST_RDY_N at any time. It does not need to wait for the source ready signal to be asserted before asserting its own ready signal and, therefore, it can assert DST_RDY_N first.

When the SRC_RDY_N and DST_RDY_N signals are both asserted, there is mutual acknowledgment by the source and destination that both are taking action based on the current state of the bus. The operation of these signals is described further in Table 2-5 and in Section 2.4.2 Data Flow, page 16 and Section 2.4.3 Frame Control Signaling, page 16.

Table 2-5: **Source and Destination Ready Signals**

Signal Name	Direction	Description
SRC_RDY_N	src to dst	**Source Ready:** Indicates that the source is ready to transact a data or control beat. No data can be transferred and no control signal can be acted upon unless both the source and the destination are ready.
DST_RDY_N	dst to src	**Destination Ready:** Indicates that the destination is ready to transact a data or control beat. No data can be transferred and no control signal can be acted upon unless both the source and the destination are ready.

2.4.2 Data Flow

Source ready (SRC_RDY_N) and destination ready (DST_RDY_N) are used to regulate the flow of data from the source to the destination (see waveforms in Section 3.5 Frame with Source and Destination Flow Control, page 37.

2.4.2.1 Source Data Flow

Source ready (SRC_RDY_N) is asserted by the source, when it is ready to transfer data. At any point in the frame transfer, the source can deassert ready for any reason (for example, the source temporarily runs out of data), which stalls the current transfer to the destination.

2.4.2.2 Destination Data Flow

Destination ready (DST_RDY_N) is asserted when destination is ready to accept data. This can be before or after it has detected that the source interface has asserted source ready (SRC_RDY_N). The destination can deassert destination ready (DST_RDY_N) at any point if it temporarily cannot accept data, causing a data stall.

2.4.3 Frame Control Signaling

The LocalLink interface uses four signals (SOF_N, SOP_N, EOP_N, and EOF_N) to control the framing of user data (see Section 2.5.1 Framing Control Signals, page 17).

The signals SRC_DSC_N and DST_DSC_N support frame discontinuation (see Section 2.6 Source and Destination Discontinue (Optional), page 26). The role of SRC_RDY_N and DST_RDY_N in qualifying these signals is described in the following sections.

2.4.3.1 Framing Signal Qualification

When the source asserts any of the four framing signals, the destination must ignore it unless SRC_RDY_N is also asserted. When any of the framing signals is asserted along with SRC_RDY_N *and* the destination asserts DST_RDY_N, a control beat is transacted.

2.4.3.2 Discontinue Signaling

When SRC_DSC_N is asserted, the destination must ignore it unless SRC_RDY_N is also asserted. If DST_RDY_N is not asserted, no discontinue action is taken.

When DST_DSC_N is asserted, the source must ignore it unless DST_RDY_N is also asserted. If SRC_RDY_N is not asserted, no discontinue action is taken.

For detail on *discontinue signaling*, see Section 2.6 Source and Destination Discontinue (Optional), page 26.

2.4.3.3 Reset

Reset is the only exception to the *both readies* rule because it does not cause a bus cycle. The reset (RST_N) signal is acted upon without regard to the state of the ready signals.

2.5 Frame Transfer Mechanisms

The LocalLink interface supports frame-based data transfers. The frame transfer methods are described in this section.

2.5.1 Framing Control Signals

The frame transfer method uses four framing control signals (SOF_N, SOP_N, EOP_N, and EOF_N) to delineate the three possible elements (header, payload, and footer) of a frame. Frame-based transfers are delineated with SOF_N (start-of-frame) and EOF_N (end-of-frame); depending on the particular application, SOP_N (start-of-payload) and/or EOP_N (end-of-payload) signals can also be specified. The definition and direction of each of the framing control signals are shown in Table 2-6.

The source ready (SRC_RDY_N) and destination ready (DST_RDY_N) signals must both be asserted for a control beat to occur. Once a control beat occurs, the framing signal *must* be deasserted.

These framing control signals are used in conjunction with the remainder buses discussed in Section 2.5.2 Remainder Buses (Optional), page 18. See Section 3 Timing Diagrams, page 31 for examples of use.

Table 2-6: **Framing Control Signals**

	Signal Name	Direction	Description
Required	SOF_N	src to dst	**Start-of-Frame:** Indicates a beat that contains the first byte(s) of a frame. If used with the remainder bus, the first byte of the frame can be any byte in the beat. If the remainder bus is already being used during the beat, REM_SOF is used instead. When used without a remainder bus, it is assumed the frame starts at the leftmost byte of the beat.
	EOF_N	src to dst	**End-of-Frame:** Indicates a beat that contains the last byte(s) of a frame. The EOF_N signal is always used with the remainder bus, unless the position of the last byte is unambiguous. The remainder bus points to the last byte in the frame.
Optional	SOP_N	src to dst	**Start-of-Payload:** Indicates a beat that contains the first payload byte(s) of a frame, if the frame has a header. If used with the remainder bus, the first byte of the payload can be any byte in the beat. If the remainder bus is already being used during the beat, REM_SOP is used instead. When used without a remainder bus, it is assumed that the payload starts at the leftmost byte of the beat.
	EOP_N	src to dst	**End-of-Payload:** Indicates a beat that contains the last payload byte(s) of a frame, if the frame has a footer. If used with the remainder bus, the last byte of the payload can be any byte in the beat. If the remainder bus is already being used in the beat, REM_EOP is used instead. When used without a remainder bus, it is assumed that the payload ends at the rightmost byte of the beat.

2.5.2 Remainder Buses (Optional)

The LocalLink interface defines remainder buses to allow frame elements to begin and end at any byte of the user interface. During a control beat, the remainder buses indicate the boundary between two parts of a frame within the control beat.(1)

Remainder buses are optional. For example, if in a particular implementation, frame data always uses all the bytes in a beat, then remainder buses are not needed. Remainder buses are used only as needed to point to the beginning or end frame elements that are not otherwise discernible. If partial beats are used in an implementation, say at the end of a frame, then a remainder bus is needed to point to the last byte of the frame.

The general rule is, if the beginning and end of all frame elements can be found:

- Without the aid of a remainder bus, then no remainder bus is needed
- With the aid of one remainder bus, then only one remainder bus is needed
- With the aid of two remainder buses, then only two remainder buses are needed

The remainder buses can be implemented as an encoded value bus or as a mask as described in Table 2-7.

Table 2-7: **Remainder Bus Types**

Type	Bus Range	Description
Encoded	*rem_bus_name*[*p*:0] or *rem_bus_name*[0:*p*] *p*=ceiling of: $\{\log_2(d/8)\}-1$	Uses an encoded value that points to a specific byte in a word to delineate the boundary between multiple frame elements (inter-frame gap, header, payload, footer) that occur in a control beat accompanied by one or more of the four framing controls (SOF_N, SOP_N, EOP_N, or EOF_N). See Section 2.5.2.1 Encoded Value Remainder Buses, page 21 for mapping of the remainder bus values.
Mask	*rem_bus_name*[*p*:0] or *rem_bus_name*[0:*p*] *p*=(*d*/8)–1	Uses a bit-per-byte mask to delineate the boundary between multiple frame elements (inter-frame gap, header, payload, footer) that occur in a data beat accompanied by one of the four framing controls (SOF_N, SOP_N, EOP_N, or EOF_N). The mask type remainder can be implemented as active-High or active-Low. See Section 2.5.2.3 Mask-Type Remainder Buses, page 23 for mapping of the remainder bus bits.

Note: *d* is the number of data bits and is a multiple of 8.

1. Because the remainder buses are undefined (invalid), except during control beats, the implementor can convey sideband information on the bus when there are no LocalLink control signals asserted.

Four remainder (REM) buses are defined below:

1. **REM:** If required, the REM bus typically accompanies EOF_N signaling to delineate the *end* of frame data, separating it from the beginning of the frame bytes or inter-frame gap (IFG) that follow. However, the REM bus can be used in conjunction with any of the four framing control signals to delineate the boundary between any two frame elements. The other three uses are as follows:
   - During an SOF_N control beat, the REM bus specifies the start of a frame
   - During an SOP_N control beat, the REM bus specifies the start of the payload portion of a frame
   - During an EOP_N control beat, the REM bus specifies the end of the payload portion of a frame

If a particular implementation requires that more than two frame elements be delineated in a single beat, the implementor uses REM_SOF, REM_SOP, and/or REM_EOP as defined below:

2. **REM_SOF:** If required, this bus is used during a beat that has SOF_N asserted to delineate the *start* of frame bytes, either header bytes or payload bytes thus separating them from the end of the preceding frame or IFG.
3. **REM_SOP:** If required, this bus is used during a beat that has SOP_N asserted to delineate the *start* of payload bytes separating it from the end of the preceding header.
4. **REM_EOP:** If required, this bus is used during a beat that has EOP_N asserted to delineate the *end* of payload bytes, separating it from the beginning of the footer bytes that follow.

The LocalLink destination only evaluates a remainder bus when its corresponding framing signal is active (see Table 2-8).

Table 2-8: **Remainder Bus and Corresponding Framing Control Signals**

Remainder Bus	Corresponding Framing Control Signal
REM_SOF (start-of-frame remainder)	SOF_N (start-of-frame)
REM_SOP (start-of-payload remainder)	SOP_N (start-of-payload)
REM_EOP (end-of-payload remainder)	EOP_N (end-of-payload)
REM (end-of-frame remainder or general purpose remainder)	EOF_N (end-of-frame) If REM is the only remainder bus used in an implementation, it may be used in conjunction with any or all of the framing control signals. If the other remainder buses are used, this bus is typically associated with EOF_N (end-of-frame).

The description and direction information for the remainder buses is summarized in Table 2-9. For examples of remainder bus usage, see sections Section 3.1 Frame with Encoded Remainder Bus, page 31, Section 3.2 Frame with Header and Mask Remainder Bus, page 32, and Section 3.3 Frame with Header and Footer, page 34.

Table 2-9: **Remainder Buses**

	Signal Name	Direction	Description
Optional	REM[*p*:0] or REM[0:*p*]	src to dst	**End-of-Frame Remainder Bus:** Used primarily to indicate the position of the last byte of a frame in a word that is accompanied by an asserted EOF_N. If in a particular implementation, the beginning and end of all frame elements is always *unambiguous* without the aid of a remainder bus, then no remainder bus is needed. If the beginning and end of all frame elements is *ambiguous*, then one or more remainder buses must be used.
	REM_SOF[*p*:0] or REM_SOF[0:*p*]	src to dst	**Start-of-Frame Remainder Bus:** Indicates the position of the first byte of a frame in a word that is accompanied by an asserted SOF_N.
	REM_SOP[*p*:0] or REM_SOP[0:*p*]	src to dst	**Start-of-Payload Remainder Bus:** Indicates the position of the first payload byte in a word that is accompanied by an asserted SOP_N.
	REM_EOP[*p*:0] or REM_EOP[0:*p*]	src to dst	**End-of-Payload Remainder Bus:** Indicates the position of the last payload byte in a word that is accompanied by an asserted EOP_N.

Note: Where d is the number of data bits and is a multiple of 8:
If encoded value REM bus, then p=ceiling{$\log_2(d/8)$}–1
If mask type REM bus, then p=(d/8)–1

2.5.2.1 Encoded Value Remainder Buses

If the encoded value method is used, the remainder bus value that accompanies a SOF_N or SOP_N control beat points to the position of the *first* byte of a frame or payload, respectively. The remainder bus value that accompanies either EOP_N or EOF_N control beat points to the position of the *last* byte of a payload or frame, respectively. Numbering of the bytes always begins with the leftmost byte and is always numbered zero.

Table 2-10, Table 2-11, Table 2-12, and Table 2-13 show examples of the four usages. Note that in the four examples below only one framing signal is asserted. See Section 2.5.2.2 Multiple Encoded Value Remainder Buses, page 22 for an example of control beats with multiple framing signals asserted.

Table 2-10 shows an 8-byte example of a start-of-frame control beat with the frame starting in the middle of the word. As shown, this case requires the encoded remainder bus value to be 4, indicating that the first byte of the frame is DATA[31:24].

Table 2-10: **SOF_N Control Beat**

remainder bus value = 4	0	1	2	3	4	5	6	7
Data Bus (MSB)	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
	<---- Previous Frame or IFG ---->				<--- Header or Payload Bytes --->			

Table 2-11 shows an 8-byte example of the transition from header bytes to payload bytes in the middle of the word. As shown, this case requires the encoded REM bus value to be 4, indicating that the first byte of the payload is DATA[31:24]. This transition can occur at any byte position. A REM bus value of 0, for example, indicates that payload bytes start at the leftmost byte.

Table 2-11: **SOP_N Control Beat**

REM bus value = 4	0	1	2	3	4	5	6	7
Data Bus (MSB)	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
	<---------- Header Bytes ---------->				<---------- Payload Bytes ---------->			

Table 2-12 shows an 8-byte example of the transition from payload bytes to footer bytes in an EOP_N beat. Frames that do not have a footer use an EOF_N framing signal instead of an EOP_N framing signal.

Table 2-12: **EOP_N Control Beat**

REM bus value = 5	0	1	2	3	4	5	6	7
Data Bus (MSB)	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
	<---------- Payload Bytes ---------->						Footer Bytes	

Table 2-13 shows an 8-byte interface example of an end-of-frame control beat. In this case, the REM bus value of 4 indicates that the last byte of the frame is located at DATA[31:24].

Table 2-13: **EOF_N Control Beat**

REM bus value = 4	0	1	2	3	4	5	6	7
Data Bus (MSB)	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
	<---------- Payload or Footer Bytes ---------->					<- Next Frame or IFG ->		

2.5.2.2 Multiple Encoded Value Remainder Buses

If more that two frame elements exist in a single control beat, then the implementer can use REM_SOF, REM_SOP and/or REM_EOP, in addition to REM, in the same control beat.

Table 2-14 shows the use of the encoded forms of the remainder buses all in one control beat.

In the example REM_SOF and REM_SOP are used to delineate the beginning and end of the header portion of the frame. The REM_SOF value points to the beginning of the header at DATA[79:72]. The last header byte is DATA[71:64] because the value of REM_SOP indicates that DATA[63:56] is the first byte of the payload.

The REM_SOP and REM_EOP buses are used to delineate the beginning and end of the payload portion of the frame. The REM_SOP bus points to the start-of- payload at DATA[63:56] and REM_EOP points to the last byte of the payload at DATA[31:24].

The footer is delineated by REM_EOP and REM. The first byte of the footer is DATA[23:16] because the value of REM_EOP indicates that DATA[31:24] is the last byte of the payload. The REM bus points to the last byte of the frame and therefore the last byte of the footer at DATA[15:8].

Table 2-14: **Multiple Encoded Value REM Bus Use**

REM_SOF value	0	1	2	3	4	5	6	7	8	9	10	11
REM_SOP value	0	1	2	3	4	5	6	7	8	9	10	11
REM_EOP value	0	1	2	3	4	5	6	7	8	9	10	11
REM value	0	1	2	3	4	5	6	7	8	9	10	11
Data Bus (MSB)	95:88	87:80	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
	IFG		Header Bytes		<---------- Payload Bytes ---------->					Footer Bytes		IFG

2.5.2.3 Mask-Type Remainder Buses

If the mask method is used, the pattern of the mask during a control beat indicates which bytes belong to a particular frame element.

Table 2-15, Table 2-16, Table 2-17, and Table 2-18 show examples of the four usages. Note that in the four examples below only one framing signal is asserted and an active-Low mask is used. See Section 2.5.2.4 Multiple Mask-Type Remainder Buses, page 24 for examples of control beats with multiple framing signals asserted.

Table 2-15 shows an 8-byte example of a start-of-frame control beat with the frame starting in the middle of the word. As shown, this case requires the REM bus mask to be asserted at byte 4, indicating that the first byte of the frame is DATA[31:24].

Table 2-15: **SOF_N Control Beat**

REM bus mask = F0h	1	1	1	1	0	0	0	0
Data Bus (MSB)	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
	Previous Frame or IFG				Header or Payload Bytes			

Table 2-16 shows an 8-byte example of the transition from header bytes to payload bytes in the middle of the word. As shown, this case requires the REM bus mask to be asserted at byte 4, indicating that the first byte of the payload is DATA[31:24]. This transition can occur at any byte position. A REM bus mask of 00h indicates that payload bytes start at the leftmost byte.

Table 2-16: **SOP_N Control Beat**

REM bus mask = F0h	1	1	1	1	0	0	0	0
Data Bus (MSB)	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
	Header Bytes				Payload Bytes			

Table 2-17 shows an 8-byte example of the transition from payload bytes to footer bytes in an EOP_N beat. Frames that do not have a footer use an EOF_N framing signal instead of an EOP_N framing signal.

Table 2-17: **EOP_N Control Beat**

REM bus mask = 03h	0	0	0	0	0	0	1	1
Data Bus (MSB)	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
	Payload Bytes						Footer Bytes	

Table 2-18 shows an 8-byte interface example of an end-of-frame control beat. In this case, the REM bus mask indicates that the last byte of the frame is located at DATA[31:24].

Table 2-18: **EOF_N Control Beat**

REM bus mask = 07h	0	0	0	0	0	1	1	1
Data Bus (MSB)	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
	Payload or Footer Bytes					Next Frame or IFG		

2.5.2.4 Multiple Mask-Type Remainder Buses

If more that two frame elements exist in a single control beat, then the implementor can use REM_SOF, REM_SOP, and/or REM_EOP, in addition to REM, in the same control beat. Table 2-19 demonstrates the use of the REM mask and the REM_EOP mask during a control beat that is accompanied by both the end-of-payload and end-of-frame control signals. The REM_EOP mask is asserted for the nine most-significant bytes indicating that the nine most-significant bytes are the end of the payload. In this case, the REM mask is asserted for bytes [23:16] and [15:8] indicating that those bytes are footer bytes.

Table 2-19: **Control Beat with Two Mask Remainders at the End of a Frame**

REM_EOP mask	0	0	0	0	0	0	0	0	0	1	1	1
REM mask	1	1	1	1	1	1	1	1	1	0	0	1
Data Bus (MSB)	95:88	87:80	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
	Payload Bytes									Footer Bytes		IFG

Table 2-20 demonstrates the use of the REM_SOF mask along with the REM_SOP mask during a control beat that is accompanied by both start-of-frame and start-of-payload control signals. Note, in this example, the most-significant byte is not part of a frame. The REM_SOF mask is asserted for bytes [87:80], [79:72], and [71:64] indicating the position of the three header bytes. The REM_SOP mask is asserted for the remaining bytes indicating the position of the first eight bytes of the payload.

Table 2-20: **Control Beat with Two Mask Remainders at the Beginning of a Frame**

REM_SOF mask	1	0	0	0	1	1	1	1	1	1	1	1
REM_SOP mask	1	1	1	1	0	0	0	0	0	0	0	0
Data Bus (MSB)	95:88	87:80	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
	IFG	Header Bytes			Payload Bytes							

Table 2-21 shows an entire frame in a single beat (no footer). In this case, the beat is accompanied by three frame control signals: start-of-frame, start-of-payload, and end-of-frame. These three signals indicate that the three remainder buses contain valid masks. Note, in this case, the REM_SOP mask and the REM mask are the same.

Table 2-21: **Control Beat with Three Mask Remainders (no REM_EOP)**

REM_SOF mask	1	1	0	0	1	1	1	1	1	1	1	1
REM_SOP mask	1	1	1	1	0	0	0	0	0	1	1	1
REM mask	1	1	1	1	0	0	0	0	0	1	1	1
Data Bus (MSB)	95:88	87:80	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
	IFG		Header Bytes		Payload Bytes					IFG		

Table 2-22 demonstrates the use of all four of the remainder buses simultaneously in the same control beat. In the example, the REM_SOF mask is CFFh indicating that header bytes begin with byte [79:72] and end with byte [71:64]. In this example there are both a REM_SOP mask and a REM_EOP mask. When this occurs the masks are the same, F07h in this case, and indicate that the payload bytes begin with byte [63:56] and end with byte [31:24]. The REM mask is FF9h indicating that footer bytes begin with byte [23:16] and end with byte [15:8].

Table 2-22: **Control Beat with Four Mask Remainders in a Single Data Beat**

REM_SOF mask	1	1	0	0	1	1	1	1	1	1	1	1
REM_SOP mask	1	1	1	1	0	0	0	0	0	1	1	1
REM_EOP mask	1	1	1	1	0	0	0	0	0	1	1	1
REM mask	1	1	1	1	1	1	1	1	1	0	0	1
Data Bus (MSB)	95:88	87:80	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
	IFG		Header Bytes		Payload Bytes					Footer Bytes		IFG

2.6 Source and Destination Discontinue (Optional)

The source discontinue (SRC_DSC_N) signal enables the source interface to indicate that the frame currently being transferred should be aborted. The precise interpretation of a source discontinue and the action the destination should take is implementation dependent, but it normally indicates the frame has an error and should be dropped. The source can use the discontinue to prematurely end a frame transfer. The source interface is required to assert end-of-frame (EOF_N) concurrently with source discontinue to end the transfer. Both the source ready and the destination ready signals must be asserted for the source discontinue control beat to be executed; otherwise, the source interface must keep source discontinue and end-of-frame asserted. The signal definition and direction is shown in Table 2-23 and examples of usage are shown in Section 3.6 Frame with Source Discontinue, page 38, and Section 3.7 Frame with Destination Discontinue, page 39.

If the source interface implements concurrent frame transfers, then a source discontinue applies to any frame that ends in the same cycle. If multiple frames end in the cycle (that is, one or several frames are contained within a word), then the source discontinue invalidates all the frames ending in that cycle.

The destination discontinue (DST_DSC_N) signal enables the destination interface to indicate that the frame currently being transferred will be aborted. The precise interpretation of a destination discontinue and the action the source should take is implementation dependent, but it normally indicates the destination has run out of buffer space and the frame should be resent. The destination can assert a destination discontinue at any point in a frame transfer. Both the source ready and the destination ready signals must be asserted for the destination discontinue control beat to be executed; otherwise, the destination interface must keep destination discontinue asserted. If the destination discontinue occurs on a cycle other than an end-of-frame, the source interface is required to prematurely end the transfer. This must be done by asserting end-of-frame (EOF_N) either the cycle after a destination discontinue control beat is executed, or several cycles later with source ready deasserted during the intervening cycles. The end-of-frame must be held asserted if both source ready and destination ready signals are not asserted. The signal definition and direction is shown in Table 2-23.

If the destination interface implements concurrent frame transfers, then a destination discontinue applies to any frame that is in progress. If multiple frames are in the cycle (that is, one or several frames are contained within a word), then the destination discontinue invalidates all the frames in that cycle (including those that start, end, and those that are completely contained within). If the destination discontinue occurs on a cycle where the frame (or last frame, in the case of multiple frames within a word) has not yet ended, the source interface is required to prematurely end the transfer. This must be done by asserting end-of-frame (EOF_N) either the cycle after a destination discontinue control beat is executed, or several cycles later with source ready deasserted during the intervening cycles. The end-of-frame must be held if both source ready and destination ready signals are not asserted.

Table 2-23: **Source and Destination Discontinue Signals (Optional)**

	Signal Name	Direction	Description
Optional	SRC_DSC_N	src to dst	**Source Discontinue:** Indicates the source device is cancelling this frame due to errors or other causes.
	DST_DSC_N	dst to src	**Destination Discontinue:** Indicates that the destination device is refusing to accept this frame due to insufficient buffer space or other causes.

2.7 Parity (Optional)

The parity signals transport odd or even parity information for each byte of the user data interface. This provides an extra level of system integrity support. The parameter SEL_PARITY (see Table 2-24) is used to select parity type. The destination interface asserts a parity error (PARITY_ERR_N) when the destination receives a parity mismatch. The definition and direction of each of these signals is shown in Table 2-25.

Table 2-24: **Parity Type Parameter**

Parameter	Size	Description
SEL_PARITY	1 bit	**Select Parity:** Set to 1 for even parity coverage and set to 0 for odd parity coverage.

Table 2-25: **Parity Error Signals**

	Signal Name	Direction	Description
Optional	PARITY[q:0] or PARITY[0:q] q=(d/8)–1	src to dst	**Parity Bus:** Indicates parity calculated over entire data bus on a byte basis. Provides even or odd byte parity coverage (depending on the setting of the SEL_PARITY parameter) for data presented on the data bus.
	PARITY_ERR_N	dst to src	**Parity Error:** Indicates that a parity error was detected during the previous data beat.

2.8 Channelization (Optional)

LocalLink channelization satisfies system interface requirements that define separate logical pathways within a single physical link. A channel at the source or destination interface can transport frames belonging to a class of traffic, a priority level, or a flow. User applications can receive or transmit frames on specified channels. Note, channel state is not stored by the LocalLink interface.

Table 2-26 lists the definition and direction of each signal. See sections Section 3.8 Channelized Frame, page 40, Section 3.9 Channelized Frame with Channel Ready Signaling, page 41, and Section 3.10 Channelized Frame with Zero-Cycle Channel Turnaround, page 42 for usage examples.

Table 2-26: **Channelization Signals**

	Signal Name	Direction	Description
Optional	CH[p:0] or CH[0:p] p=ceiling [{$\log_2(n)$}–1]	If src = output, then dst = input If src = input, then dst = output	**Channel Number Bus:** Used for implementations that allow read and/or write access to a specific channel. This binary encoded vector indicates which channel a data transfer request or response is coming from or going to.
	Note: If src sources CH, then dst sources CH_RDY_N; if dst sources CH, then src sources CH_RDY_N		
	CH_RDY_N[n–1:0] or CH_RDY_N[0:n–1]	If src = output, then dst = input If src = input, then dst = output	**Channel Ready Bus:** This vector indicates frame availability on the channels. The source indicates channels that have one or more frames available to send, or the destination indicates channels that can receive one or more frames. Bits corresponding to all eligible channels shall be asserted.
	CH_SWAP_LOOKAHEAD_N[n–1:0] or CH_SWAP_LOOKAHEAD_N[0:n–1] or CH_SWAP_LOOKAHEAD_N	If src = output, then dst = input If src = input, then dst = output	**Channel Swap Lookahead Bus:** When asserted, the driving interface indicates that the current channel will go into an underflow state *soon*. (*Soon* is defined by the parameter CH_LOOKAHEAD_CLK_CNT described in Section 2.8.3 Channel Swap Lookahead, page 29.) A bit-per-channel indication or single bit value (logical AND of all channel status indicators) is used to indicate lookahead status. Bits corresponding to the channel going empty are asserted.

Note: n is the number of channels

2.8.1 Channel

Channel (CH) indicates the channel affiliation of the data transfer request or response. Channel (CH) can be either:

- Source interface output, indicating channel affiliation of frame sourced, or
- Destination interface output, indicating channel affiliation of frame requested

2.8.2 Channel Ready

Channel ready (CH_RDY_N) indicates frame availability on various channels supported. Channel ready (CH_RDY_N) can be either:

- Source interface output, indicating frame availability on channels, or
- Destination interface output, indicating full or not full channels

2.8.3 Channel Swap Lookahead

The assertion of a bit in the channel swap lookahead bus (CH_SWAP_LOOKAHEAD) indicates that there are only a certain number of clock cycles remaining until the associated channel underflows or overflows as frames are drained from or are sourced to it. The parameter CH_LOOKAHEAD_CLK_CNT controls the number of clock cycles of warning. The application can adjust the channel swap lookahead indication to anticipate the latency associated with a channel switching operation to achieve zero-cycle channel turnaround. The CH_SWAP_LOOKAHEAD signal can be either:

- Source interface output, warning of frame underflow on channels, or
- Destination interface output, warning of frame overflow on channels

Table 2-27 defines the channel lookahead clock count parameter.

Table 2-27: **Channel Lookahead Clock Count Parameter**

Parameter	Unit	Description
CH_LOOKAHEAD_CLK_CNT	Clock cycle	**Channel Lookahead Clock Count:** Number of clock cycles before the current channel goes into an underflow or overflow state.

3 Timing Diagrams

3.1 Frame with Encoded Remainder Bus

A basic LocalLink frame transfer with an encoded remainder (REM) bus is shown in Figure 3-1. A transfer starts when the source interface asserts SOF_N and SRC_RDY_N and drives data on DATA[63:0]. In this particular scenario, DST_RDY_N is permanently asserted by the destination interface. The transfer ends when the source interface asserts EOF_N. The encoded REM bus value of 2 indicates that the three most-significant bytes, namely DATA[63:56], DATA[55:48], and DATA[47:40], contain the last payload bytes, making this a 51-byte frame transfer.

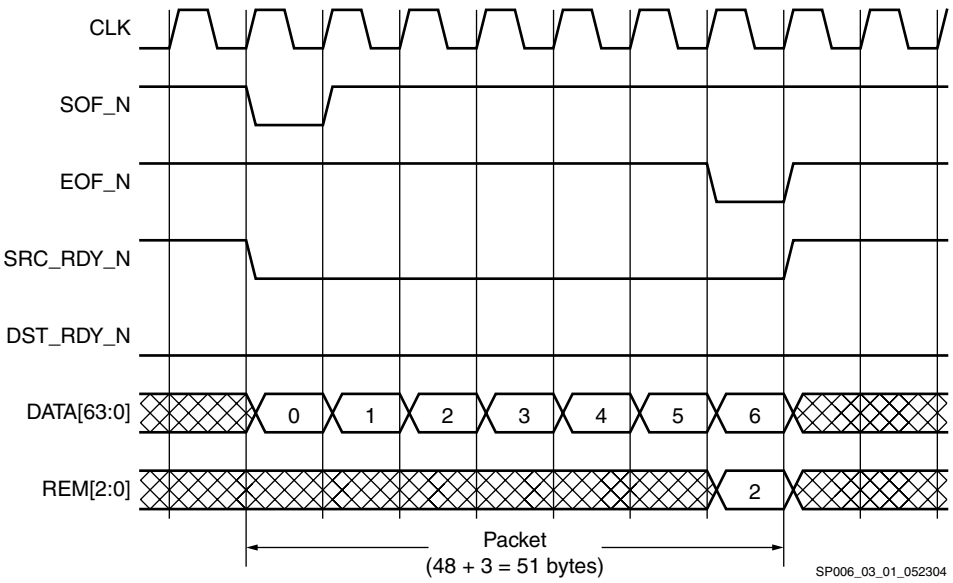

Figure 3-1: **Frame with Encoded Remainder Bus**

3.2 Frame with Header and Mask Remainder Bus

A LocalLink frame with a start-of-payload indication is shown in Figure 3-2. A transfer starts when the source interface asserts SOF_N and SRC_RDY_N and drives data on DATA[63:0]. The DST_RDY_N signal is asserted permanently by the destination interface. The source interface indicates the start-of-payload by asserting SOP_N and a remainder (REM) bus mask. This REM bus mask indicates the boundary between header bytes and payload bytes. The F0h mask indicates that during the clock cycle the four least-significant bytes, namely DATA[31:24] (the first payload byte), DATA[23:16], DATA[15:8], and DATA[7:0], contain payload bytes (see Table 3-1, page 33 for more details).

The transfer ends when the source interface asserts EOF_N. The second REM bus mask (associated with EOF_N) indicates the boundary between the end of the frame and the beginning of the next frame or an IFG. The 0Fh REM bus mask indicates that during the clock cycle, the four most-significant bytes (DATA[63:56], DATA[55:48], DATA[47:40], and DATA[39:32]) contain payload bytes; the final payload byte is DATA[39:32] (see Table 3-2, page 33 for more details).

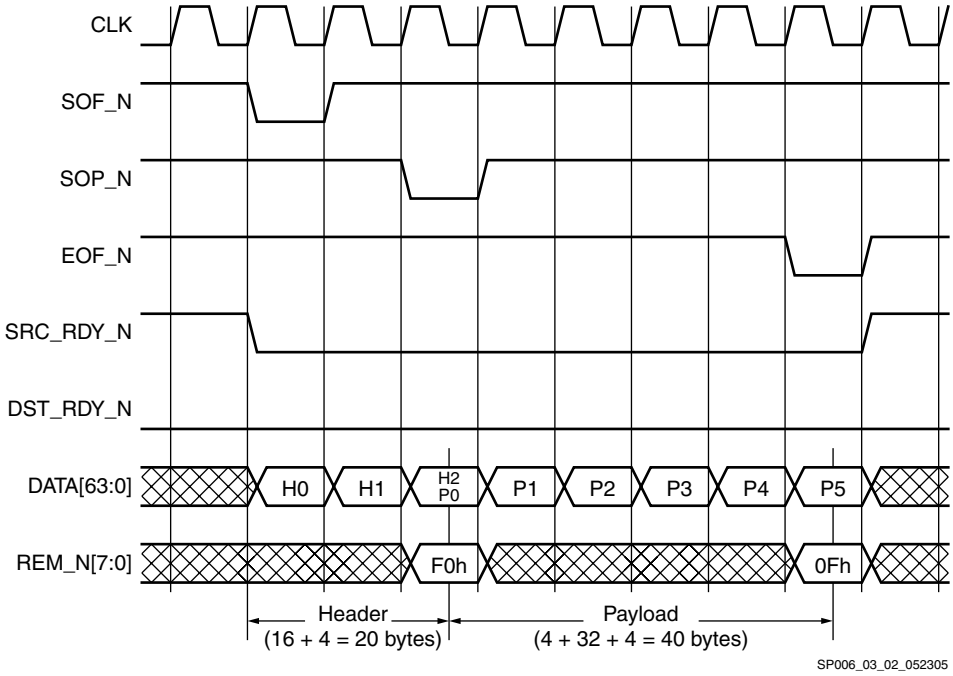

Figure 3-2: **Frame with Header and a Mask Remainder Bus**

Table 3-1: **SOP_N Beat - Mask Remainder Bus Indicates Start-of-Payload**

F					0				
MSb	1	1	1	1	0	0	0	0	LSb
MSB	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0	LSB
Header Bytes					Payload Bytes				

Table 3-2: **EOF_N Beat - Mask Remainder Bus Indicates End-of-Payload**

0					F				
MSb	0	0	0	0	1	1	1	1	LSb
MSB	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0	LSB
Payload Bytes					IFG				

3.3 Frame with Header and Footer

LocalLink frames use start-of-payload and end-of-payload frame control signals to demarcate header and footer frame elements as shown in Figure 3-3. The source interface starts the frame by asserting SOF_N and SRC_RDY_N. The DST_RDY_N signal is permanently asserted by the destination interface. The first beat transfers a word of header bytes. The second beat transfers the final four header bytes and also transfers the first four bytes of the payload.

The transition from header bytes to payload bytes occurs when the source interface asserts SOP_N. The F0h rem bus mask indicates that during the control beat the four least-significant bytes (DATA[31:24], DATA[23:16], DATA[15:8], and DATA[7:0]) contain payload bytes; the first payload byte is DATA[31:24]. See Table 3-3, page 35 for more details.

The transition from payload bytes to footer bytes occurs when the source interface asserts EOP_N. The 03h rem bus mask indicates that during the control beat the six most-significant bytes (DATA[63:56], DATA[55:48], DATA[47:40], DATA[39:32], DATA[31:24], and DATA[23:16]) contain payload bytes; the last payload byte is DATA[23:16]. The rem bus mask also indicates that the two least-significant bytes (DATA[15:8] and DATA[7:0]) contain footer bytes; the first footer byte is DATA[15:8]. See Table 3-4, page 35 for more details.

The source interface ends the transfer by asserting the EOF_N signal. The 07h rem bus mask indicates that the five most-significant bytes (DATA[63:56], DATA[55:48], DATA[47:40], DATA[39:32], and DATA[31:24]) contain footer bytes; the final footer byte, DATA[31:24], is also the final byte of the frame. See Table 3-5, page 35 for more details.

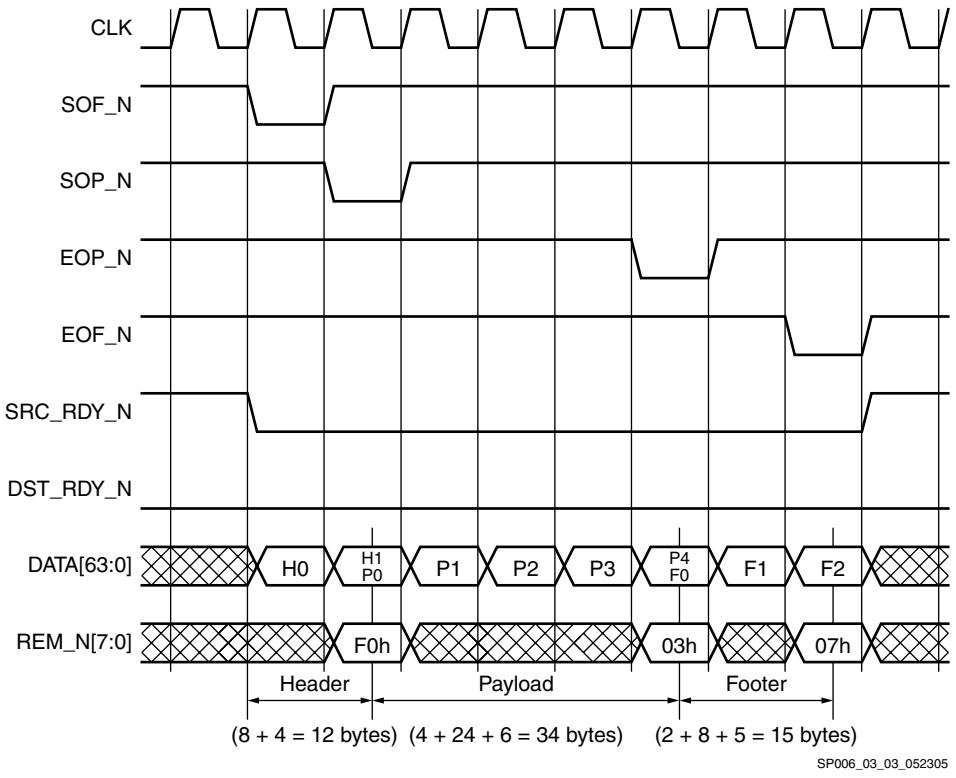

Figure 3-3: **Frame with Header and Footer**

Table 3-3, Table 3-4, and Table 3-5 show the mask rem bus detail of the SOP_N beat, EOP_N beat, and EOF_N beat, respectively.

Table 3-3: **SOP_N Beat - Mask Remainder Bus Indicates Start-of-Payload**

F					**0**				
MSb	1	1	1	1	0	0	0	0	LSb
MSB	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0	LSB
Header Bytes					Payload Bytes				

Table 3-4: **EOP_N Beat - Mask Remainder Bus Indicates End-of-Payload**

0					**3**				
MSb	0	0	0	0	0	0	1	1	LSb
MSB	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0	LSB
Payload Bytes							Footer Bytes		

Table 3-5: **EOF_N Beat - Mask Remainder Bus Indicates End-of-Frame**

0					**7**				
MSb	0	0	0	0	0	1	1	1	LSb
MSB	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0	LSB
Footer Bytes					IFG				

3.4 Back-to-Back Frames

Figure 3-4. shows two back-to-back frames that share a beat. The transfer starts when the source asserts SOF_N and SRC_RDY_N and drives data on DATA[63:0]. In this particular scenario, DST_RDY_N is permanently asserted by the destination interface. The first frame (Frame A) ends on beat 5 when the source interface asserts EOF_N and rem_n[7:0]. The rem mask is 0Fh, which indicates that the four most-significant bytes belong to Frame A.

Because SOF_N is asserted in clock cycle 5, Frame B starts in clock cycle 5; and because the four most-significant bytes belong to Frame A, Frame B starts with the remaining four bytes of the beat. Frame B ends with clock cycle 8 when EOF_N is asserted.

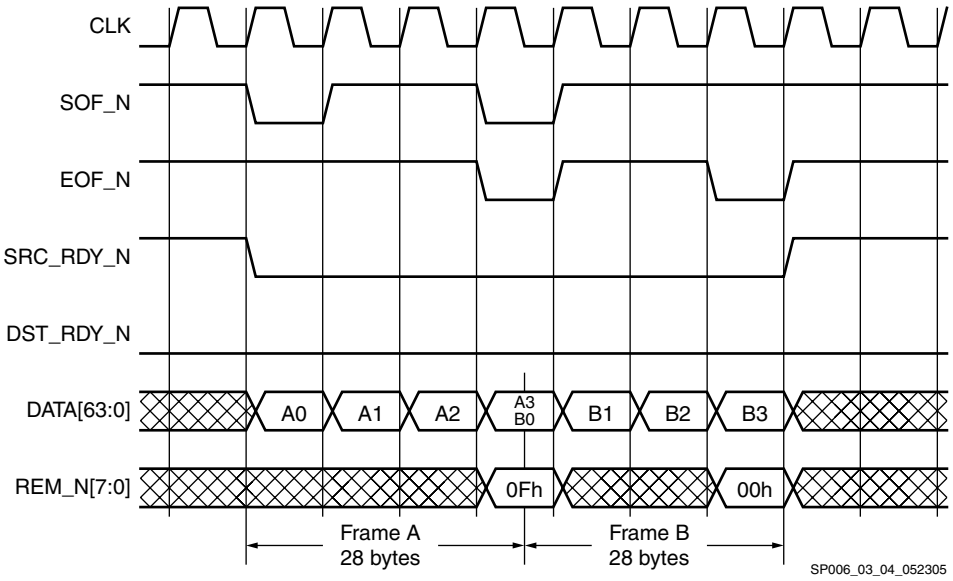

Figure 3-4: **Back-to-Back Frames**

3.5 Frame with Source and Destination Flow Control

The source and destination ready signals allow for explicit flow control on the LocalLink interface. Figure 3-5 shows a LocalLink frame transfer with source and destination flow control. The transfer starts when the source interface presents data and asserts SOF_N and SRC_RDY_N. In this example, the destination interface is not ready and holds DST_RDY_N deasserted. The source interface presents the next set of data bytes after the destination asserts DST_RDY_N. Next, the source interface deasserts SRC_RDY_N, indicating it is unable to present any new data at the next clock cycle. The transfer resumes when the source interface asserts SRC_RDY_N and presents the next data set.

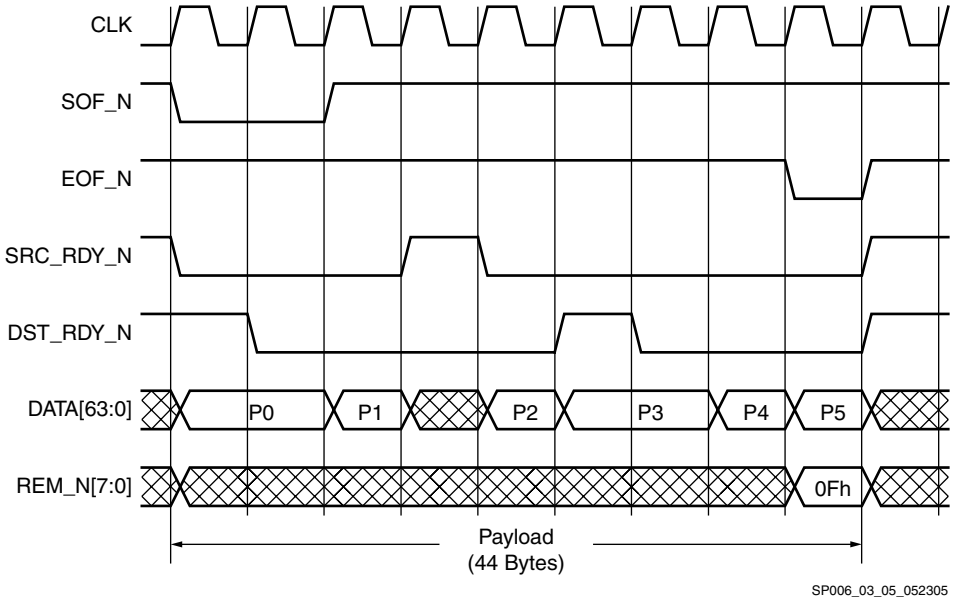

Figure 3-5: **Frame with Source and Destination Flow Control**

3.6 Frame with Source Discontinue

Figure 3-6 shows a LocalLink frame transfer where the source interface discontinues an in-process frame. The transfer starts when the source interface presents data and asserts SOF_N and SRC_RDY_N. In this example, the destination interface is ready and holds DST_RDY_N asserted. The source interface then discontinues this transfer mid-frame by asserting SRC_DSC_N and EOF_N, which signals to abort the frame transfer. The destination interface then discards the current frame data. On the next clock cycle, the source interface chooses to deassert SRC_RDY_N before beginning the new frame.

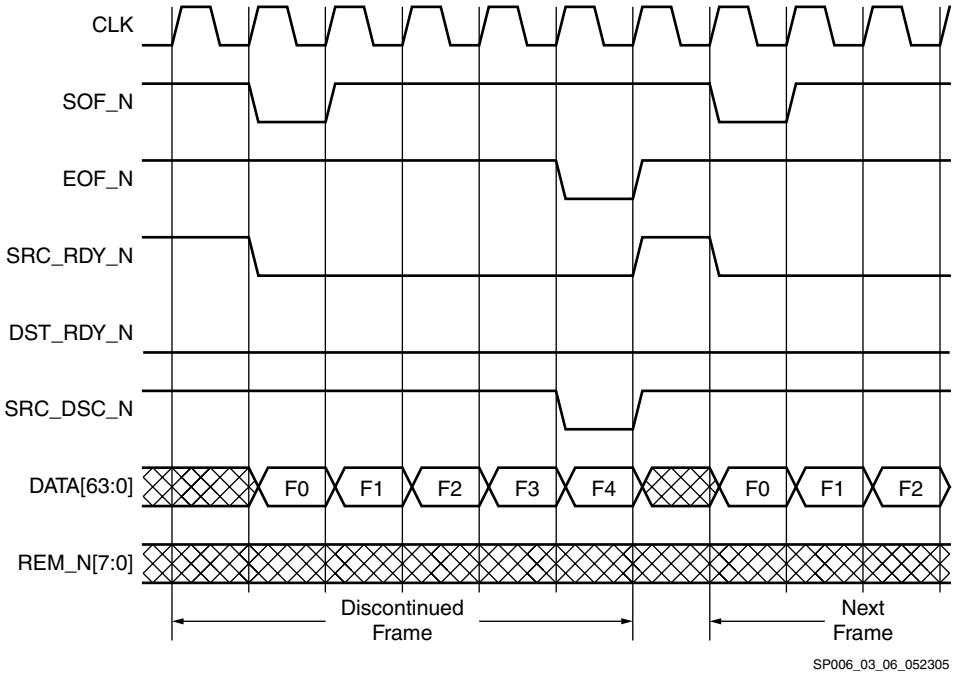

Figure 3-6: **Frame with Source Discontinue**

3.7 Frame with Destination Discontinue

Figure 3-7 shows a LocalLink frame transfer where the destination interface discontinues an in-process frame. The transfer starts when the source interface presents data and asserts SOF_N and SRC_RDY_N. In this example, the destination interface is ready and holds DST_RDY_N asserted. During the frame transfer, the destination interface asserts DST_DSC_N, which signals to the source its inability to process the current frame. The source interface is then required to assert EOF_N, discontinuing the current frame transfer.

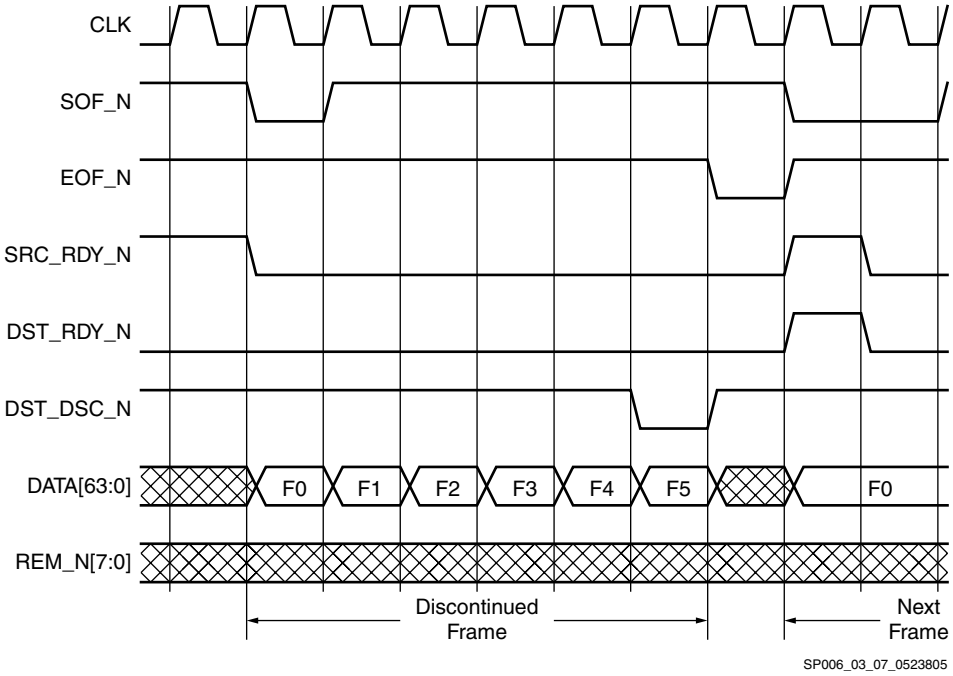

Figure 3-7: **Frame with Destination Discontinue**

3.8 Channelized Frame

Figure 3-8 shows a channelized LocalLink frame transfer. The source interface asserts SOF_N and SRC_RDY_N, and presents the first data transfer and channel number. The destination interface processes the channel request and asserts DST_RDY_N when it is ready to accept the frame transfer. The source interface can change the channel request mid-frame, which is applicable only to the subsequent frame transfer. The destination interface deasserts DST_RDY_N if it is unable to process the requested channel.

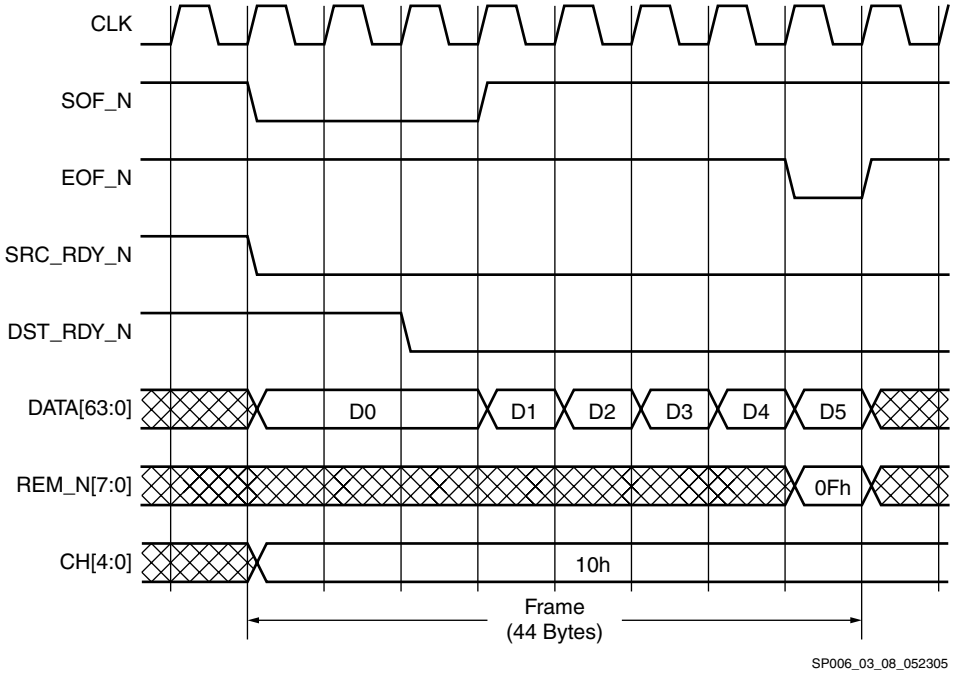

Figure 3-8: **Channelized Frame**

3.9 Channelized Frame with Channel Ready Signaling

Figure 3-9 shows an example of a channelized frame with channel ready signaling. In this 32-channel example, the CH_RDY_N bus shows that three channels can accept data. According to the value on the source's CH bus, channel 16 is responding to the CH_RDY_N bus. The four steps to transfer a channelized frame with channel ready signaling are:

1. The destination interface asserts the CH_RDY_N[31:0] bus to indicate channel availability. A typical application asserts a logic zero on the appropriate bus bits to indicate channels that can accommodate at least one full-sized LocalLink frame.
2. The source interface responds by asserting SOF_N and SRC_RDY_N, driving the data bus, and driving the number of the channel that is transferring data to the destination onto the CH[4:0] bus.
3. The destination responds by asserting DST_RDY_N.
4. The source interface ends the transfer by asserting the EOF_N signal. The 0Fh rem bus mask specifies that the four most-significant bytes (DATA[63:56], DATA[55:48], DATA[47:40], and DATA[39:32]) contain the final four bytes of the frame.

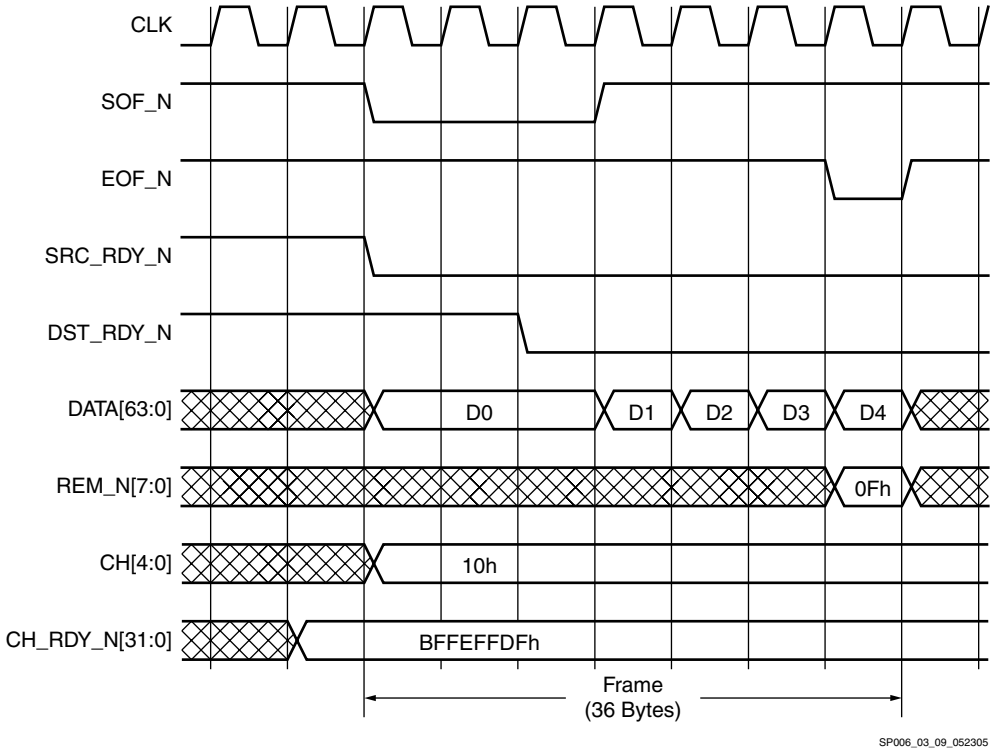

Figure 3-9: **Channelized Frame with Channel Ready Indication**

3.10 Channelized Frame with Zero-Cycle Channel Turnaround

Figure 3-10 shows a zero-cycle channel turnaround using the CH_SWAP_LOOKAHEAD_N signal driven by the source interface. The bit corresponding to a channel about to become empty is driven to a logic zero. This serves as an early warning to the destination interface regarding an impending channel change.

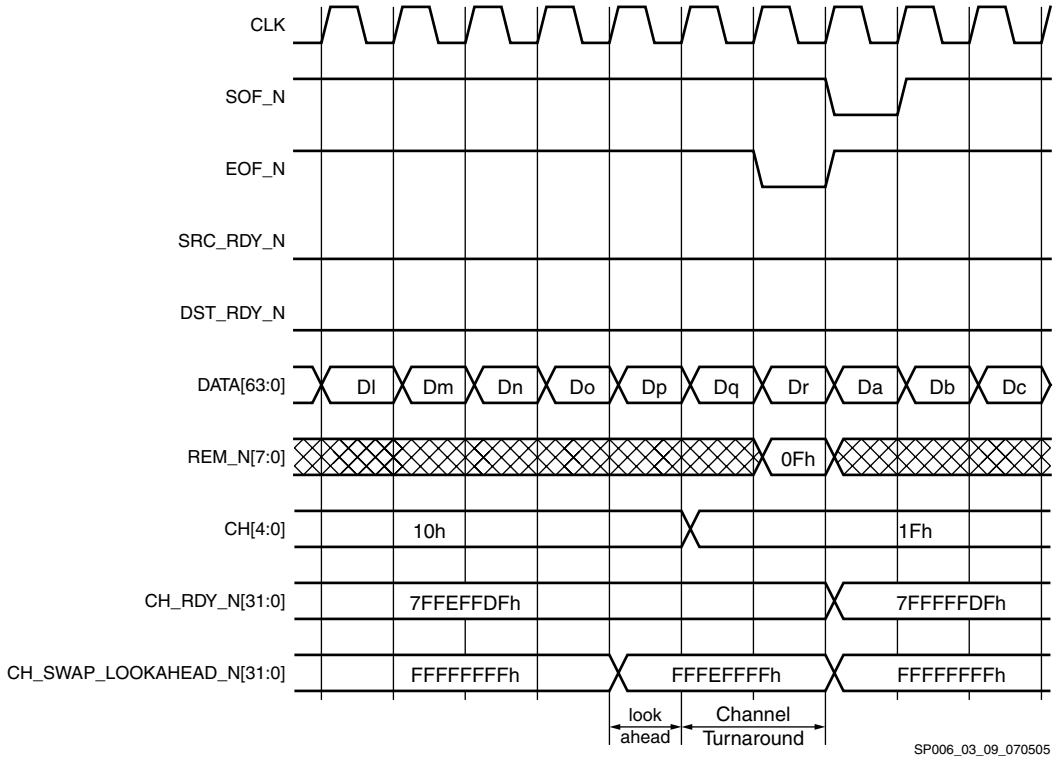

Figure 3-10: **Channelized Frame with Zero-Cycle Channel Turnaround**