Interlaken
Protocol Definition

A Joint Specification of Cortina Systems and Cisco Systems

Revision 1.2
October 7, 2008

Terms and Conditions

Cortina Systems, Inc. (“Cortina”) and Cisco Systems, Inc. (“Cisco”) (collectively, Cortina and Cisco, “We” or “Our”) desire to encourage widespread adoption and use of the Interlaken Protocol Definition set forth in the following pages (the “Specification”). Accordingly, Cortina and Cisco are willing to grant you a license to use the Specification, at no charge, subject to your agreement and compliance with the following terms and conditions. PLEASE READ THESE TERMS AND CONDITIONS CAREFULLY.

Cortina and Cisco grant you a non-exclusive, royalty-free, worldwide, perpetual license to create modifications, adaptations, translations and derivative works based on the Specification (collectively, “Modifications”), to develop products and systems implementing the Specification or implementing such Modifications, to make, use and sell all such products and systems, and to make copies of the Specification as necessary to exercise the foregoing rights. Cortina and Cisco reserve all rights not expressly granted above. For sake of clarity, no right is granted to disclose or distribute copies of this Specification to any third party. You may not disclose or distribute the Specification, unless you receive the express prior written consent of either Cortina or Cisco.

You agree to include a copyright notice on all complete and partial copies of the Specification as follows: “Copyright © 2006 Cortina Systems, Inc. and Cisco Systems, Inc.” In the event that you create Modifications of the Specification, you agree to include a notice identifying which portions of the Specifications have been modified or added by you.

In the event that you claim any intellectual property rights that are embodied in, or practiced by, any portion of the Specification (your “Related Rights”), you must agree not to enforce those intellectual property rights against Cortina, Cisco, or any licensee of the Specification, or else we are unwilling to grant you the license stated above. You hereby grant to Cortina, Cisco and all licensees of the Specification an unlimited, non-exclusive, royalty-free, worldwide, perpetual license under all of your Related Rights.

YOU ACKNOWLEDGE THAT THE SPECIFICATION IS LICENSED TO YOU ON AN “AS IS” BASIS, W ITHOUT WARRANTY OF ANY KIND. FOR SAKE OF CLARITY, CORTINA AND CISCO DISCLAIM ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE SPECIFICATION, INCLUDING W ITHOUT LIMITATION, ANY AND ALL WARRANTIES OF MERCHANTABILITY, FITNESS FOR PARTICULAR PURPOSES, TITLE AND NONINFRINGEMENT.

IN NO EVENT WILL CORTINA OR CISCO BE LIABLE TO YOU OR TO ANY THIRD PARTY FOR ANY INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION W ITH THE SPECIFICATION, AND IN NO EVENT WILL THE CUMULATIVE LIABILITY OF EACH OF CORTINA AND CISCO FOR ANY AND ALL CLAIMS IN CONNECTION WITH THE SPECIFICATION EXCEED, IN AGGREGATE, ONE HUNDRED DOLLARS (U.S. $100.00). YOU AGREE THAT THIS PARAGRAPH IS AN ESSENTIAL BASIS OF THE RIGHTS GRANTED TO YOU ABOVE, AND THAT WE ARE UNWILLING TO GRANT YOU A LICENSE IF YOU DO NOT AGREE TO THIS PROVISION.

By using the Specification in any manner, you are agreeing to these terms and conditions. If you do not agree to these terms and conditions, We do not grant you a license and you have no right to use the Specification, in which event your use of this Specification would be in violation of Cortina’s and Cisco’s intellectual property rights.

Contents

Figures

Tables

1.0 Revision History

1.1 Clarifications to Revision 1.1

Since the release of revision 1.1 a number of comments and requests for clarification have been received. Revision 1.2 is not intended to change the specification itself, but just provide more clarity where needed. Also, Appendix D (recommended implementation) has been removed as this is depreciated by the Interlaken Alliance interoperability recommendations.

Revision 1.2 contains the following clarifications:

- Updated the Burst Segmentation Algorithm Example in Section 5.3.2.1.1, *Optional Scheduling Enhancement*, on page 14.
- Updated Figure 7, *Control Word Format*, on page 16 to show the Framing Layer Control Word format as well as the Idle/Burst Control Word format.
- Updated the descriptions of the In-Band Flow Control and CRC24 fields in Table 1, *Idle/Burst Control Word Format*, on page 17.
- Updated the Figure 10, *Transmit Interface State*, on page 21 to use the standard Interlaken terminology for flow control (XON).
- Updated the Figure 11, *Out-of-Band Logical Timing Diagram*, on page 23 to show that FC_DATA represents the flow control state for specific calendar entries.
- Updated Section 5.4.2, *64B/67B Encoding*, on page 26 to document that besides positive disparity, negative disparity versions of the Block Type codes are also possible since the Interlaken protocol can bitwise invert control words as part of its disparity algorithm.
- Updated Section 5.4.2, *64B/67B Encoding*, on page 26 to specify that the algorithm used for disparity control in Interlaken guarantees the running disparity will be within +/- 96-bit bound.
- Updated Figure 16, *Scrambler Synchronization State Diagram*, on page 31 to match the text description by comparing the received Scrambler State Word to the expected scrambler state.
- Removed references to CEI-6 in the Section 5.5, *Electrical Specifications*, on page 40.

The template was updated to more closely match other Interlaken documentation, and other editorial changes were made to the specification to improve readability and add clarity.

1.2 Changes to Revision 1.0

Following the release of Version 1.0 of the Interlaken Protocol, it was identified that the scrambler polynomial and reset methodology were susceptible to a determined attack to defeat the data scrambling and introduce long run lengths of consecutive identical digits into the SerDes interconnect. To avoid pathologies associated with this behavior and to eliminate this potential, a new scrambler is chosen and a change in the reset methodology were implemented. This is the primary motivation for releasing this Version 1.1 of the Protocol. Given this opportunity, additional small changes were made to the specification to improve readability, expand Skip Word and flow control calendar usage, remove unnecessary functions, and add clarity.

Version 1.1 contains the following changes to Version 1.0:

- Change of scrambler and scrambler reset methodology, with the corresponding addition of the Scrambler State Control Word

- Definition of the Block Type as a 6-bit field in bit positions [63:58] of the Meta Frame Control Words
- Expansion of Skip Word usage to allow additional insertion of Skip Words
- Removal of the PRBS randomization of the Channel Number field of the Burst/Idle Control Words
- Elimination of the re-use option of the in-band flow control field
- Additional clarification usage of the flow control calendar, and the introduction of link-level flow control
- Addition of an introduction to the basic concepts of the interface to improve comprehension
- Addition of a table of Meta Frame Control Words
- Addition of an illustration of the new Scrambler
- Addition of a recommended implementation to assist with interoperability, and modification of the performance analysis
- Harmonization of verb tense
- Updating of references

2.0 Definitions and Key Variables

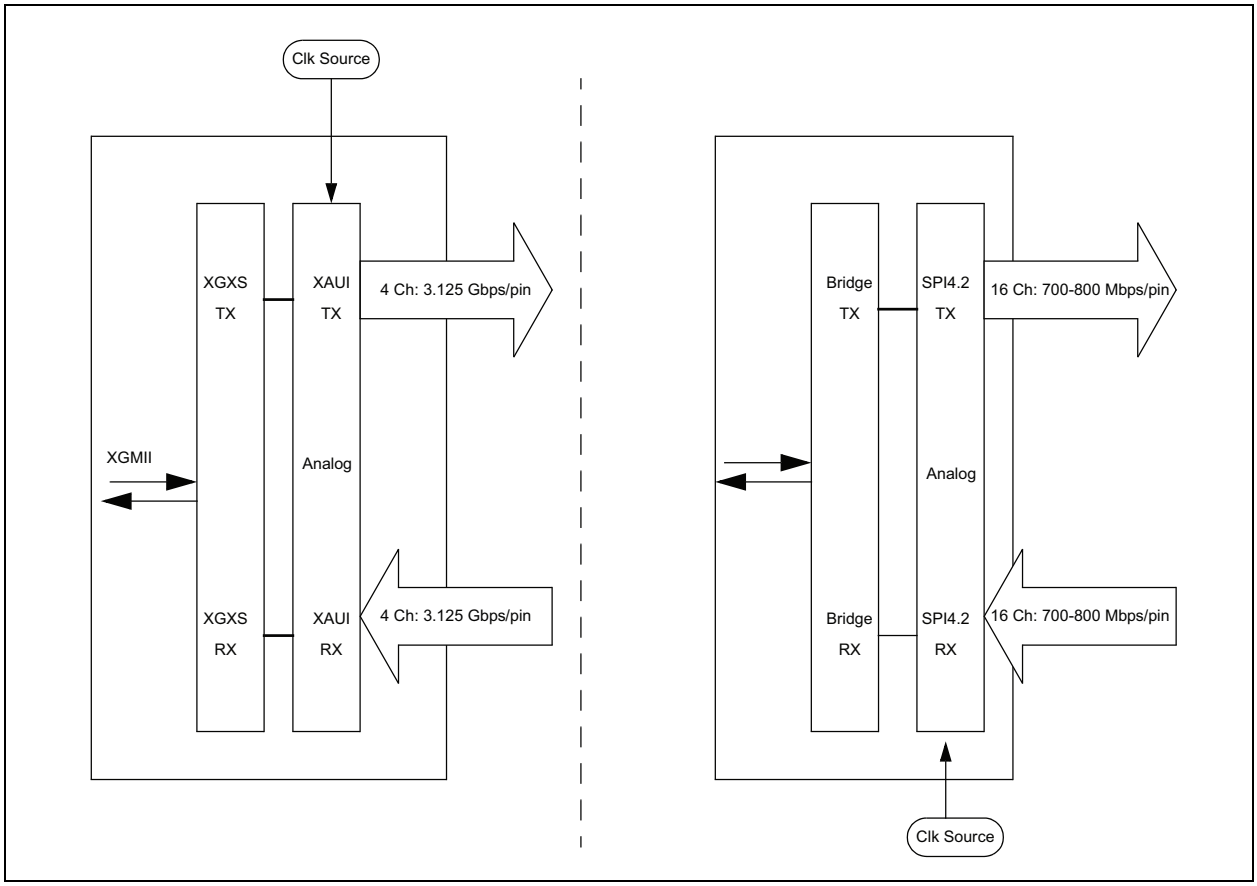
BurstMax	Maximum size of a data burst (multiple of 64 bytes)
BurstShort	Minimum interval between Burst Control Words (minimum value of 32 bytes, incrementing by 8 bytes)
BurstMin	Parameter to specify the smallest end-of-packet burst (See Section 5.3.2.1.1, *Optional Scheduling Enhancement*, on page 14)
MetaFrameLength	The quantity of data sent on each lane including one **Synchronization Word**, one **Scrambler State Word**, one **Diagnostic Word**, one or more **Skip Words**, and the data payload
Word	An 8-byte quantity, and the fundamental unit of data and control information that is transferred across the interface
Block Type	The first six bits of each Control Word, used to distinguish different types of Control Words: bits [63:58] for Synchronization, Skip, Scrambler State, and Diagnostic Words, and bit [63] for Burst/Idle Words)
Burst Control Word	A Control Word with bit 63 = ‘1’ and Type = ‘1’
Idle Control Word	A Control Word with bit 63 = ‘1’ and Type = ‘0’
Synchronization Word	A Control Word with Block Type = 0b011110 sent out on all lanes simultaneously with a periodicity of **MetaFrameLength**, used to synchronize the scrambler and perform lane alignment
Scrambler State Word	A Control Word with Block Type = 0b001010, sent immediately after the Synchronization Word, used to transmit the current scrambler state to the receiver
Skip Word	A Control Word with Block Type = 0b000111, used to provide clock compensation for repeater functions
Diagnostic Word	A Control Word with Block Type = 0b011001, sent immediately preceding the Synchronization Word, used to communicate a per-lane error diagnostic and optional per-lane status
Lane Skew Tolerance	107 UI

3.0 Introduction

The two dominant high-speed chip-to-chip interface protocols for networking applications are XAUI [1] and SPI4.2 [2]. While SPI4.2 offers important advantages in channelization, programmable burst sizes, and per-channel backpressure, the excessive width of the interface limits its scalability, and the source-synchronous nature of the protocol reduces its effective reach. Conversely, XAUI is a narrow 4-lane interface, offers long reach, and suits a variety of implementations: FR4 on PCB, backplanes, and cable. Yet as a packet-based interface it lacks channelization and flow control, restricting it from several applications. And both protocols offer only fixed configurations, limiting the ability of the designer to tailor the interface capacity to the application.

This document defines a new protocol, Interlaken, that enables the design of a narrow, high-speed, channelized packet interface.

Figure 1 XAUI Versus SPI4.2 Interfaces



4.0 Applications

Interlaken can be used in a variety of applications:

- Framer/MAC to NPU or L2/L3 switch interface
- Line Card to Switch Fabric Interface

It can also run on multiple medias: FR4 (PCB), backplanes, or over cable.

Figure 2 Framer/MAC to NPU/L2 or L3 Switch

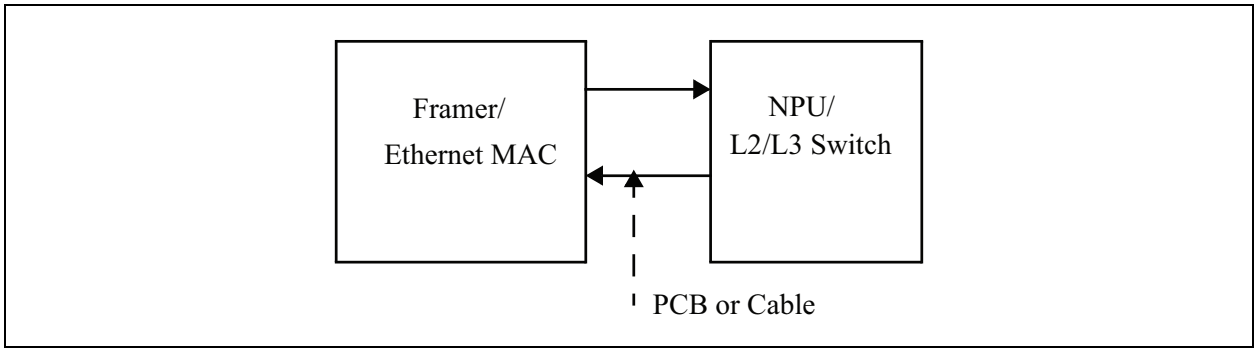
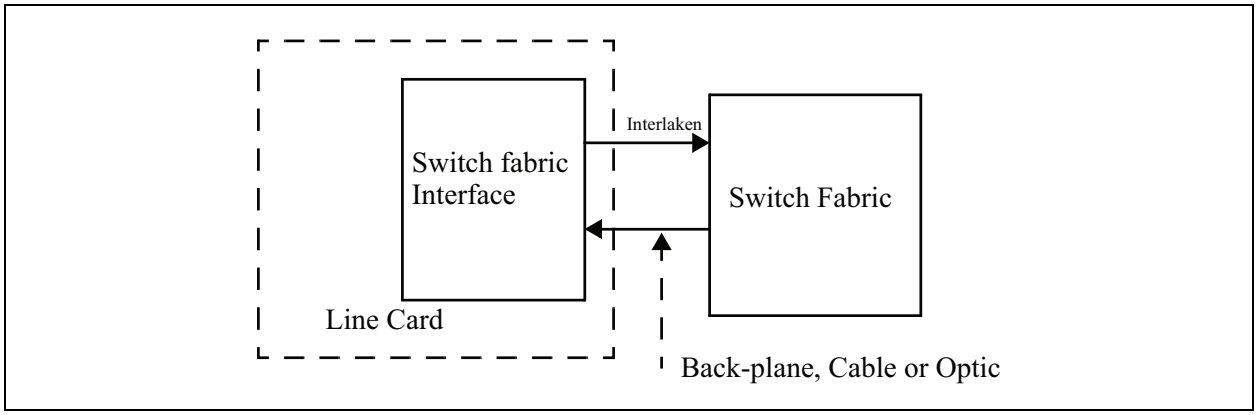


Figure 3 Framer/MAC to NPU/L2 or L3 Switch



5.0 Interlaken Protocol

5.1 Fundamentals

Interlaken is a narrow, high-speed channelized chip-to-chip interface. It is characterized by the following features:

- Support for 256 communications channels, or up to 64K with channel extension
- A simple control word structure to delineate packets, similar in function to SPI4.2
- A continuous Meta Frame of programmable frequency to guarantee lane alignment, synchronize the scrambler, perform clock compensation, and indicate lane health
- Protocol independence from the number of SerDes lanes and SerDes rates
- Both out-of-band and in-band per-channel flow control options, with a simple Xon/Xoff semantic
- 64B/67B data encoding and scrambling
- Performance that scales with the number of lanes

5.2 Basic Concepts

There are two fundamental structures that define the Interlaken Protocol: the data transmission format and the Meta Frame. The data transmission format relies significantly on the concepts of SPI4.2 [2]. Data sent across the interface is segmented into bursts, which are subsets of the original packet data. Each burst is bounded by two control words, one before and one after, and sub-fields within these control words affect either the data following or preceding them for functions like start-of-packet, end-of-packet, error detection, and others. Each burst is associated with a logical channel, which can represent a physical networking port in the system or some other logically connected stream of data. Packet data is transmitted sequentially by means of one or more bursts, and the size of the bursts is a configurable parameter. By segmenting the data into bursts, the interface allows the interleaving of data transmissions from different channels for low-latency operation.

The Meta Frame is defined to support the transmission of the data over a SerDes infrastructure. It encompasses a set of four unique control words, which are defined to provide lane alignment, scrambler initialization, clock compensation, and diagnostic functions. The Meta Frame runs in-band with the data transmissions, using the specific formatting of the control words to distinguish it from the data.

The data transmission format and Meta Frame are described in detail in the following sections.

5.3 Protocol Layer

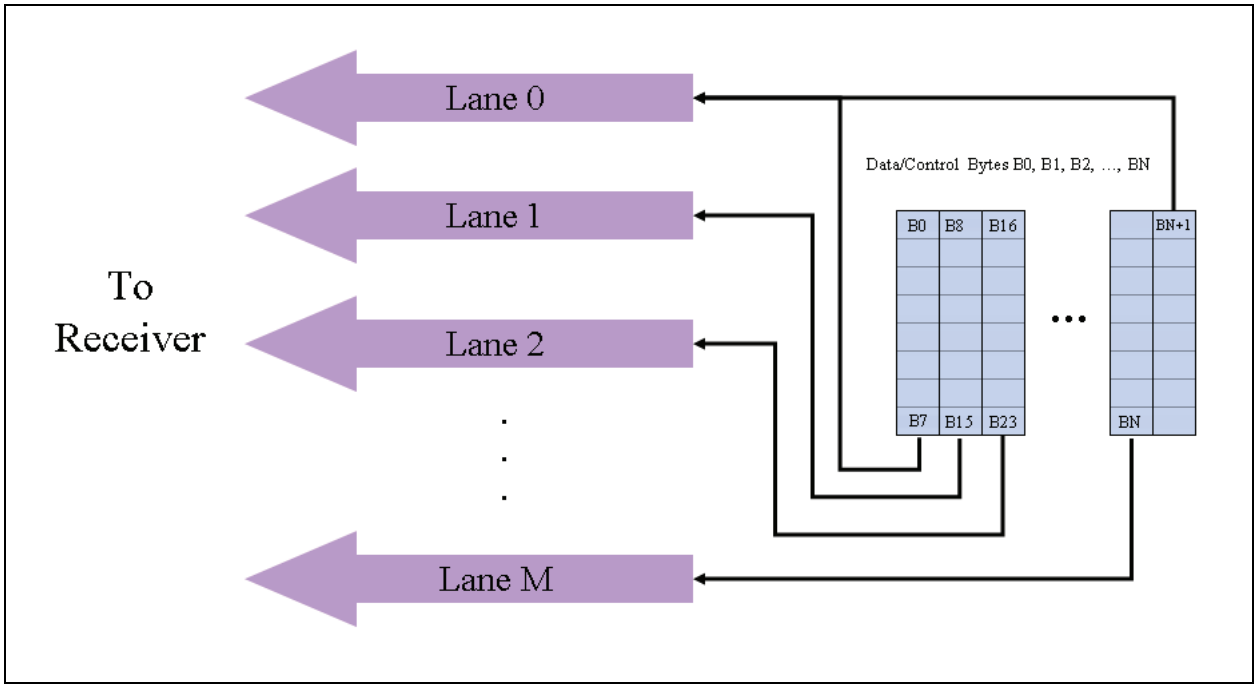
5.3.1 Transmission Format

Data is transmitted across the Interlaken interface via a configurable number of SerDes lanes. For the purpose of this document a lane is defined as a simplex serial link between two ICs. The protocol is designed to operate with any number of lanes, including only one, with no inherent maximum. Actual implementations may choose to fix their operation to a specific number of lanes; there is no requirement to support a variable number.

The fundamental unit of data sent across the interface is an 8-byte word. This number is chosen to conform to the 64B/67B encoding selected for the protocol, and is also the size of the control word used to delineate bursts. By making the fundamental transfer unit equivalent to the control word size it becomes easy to adjust the width of the interface.

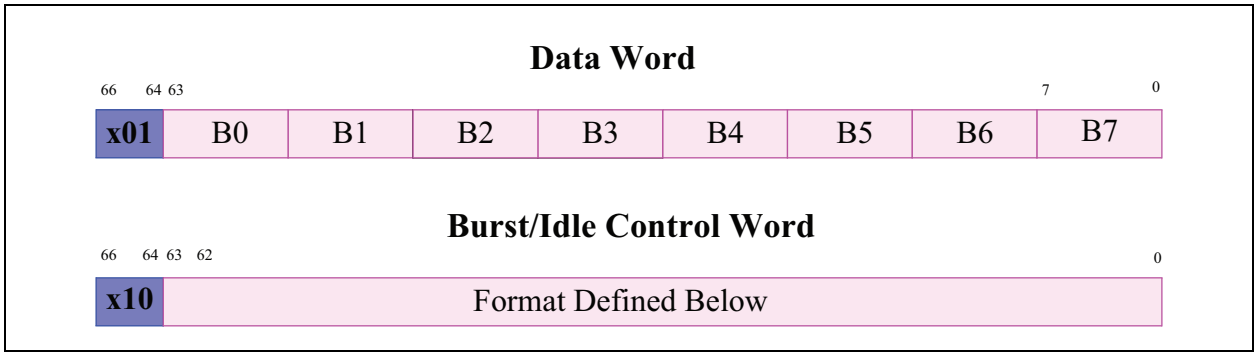
Data and control words are striped across the lanes sequentially, beginning with lane 0, ending at lane M, and repeating for the next block of data. Figure 4 illustrates the process:

Figure 4 Lane Striping Example



64B/67B encoding occurs on each lane individually. Transport is accomplished via two fundamental word types: Data Words and Burst/Idle Control Words, which are distinguished via the 64B/67B framing bits. The format of these two word types is illustrated in Figure 5 on page 13:

Figure 5 Word Formats



Both data and control information is transmitted in bit order (msb to lsb within each byte), from bit[66] through bit[0].

The Framing Layer introduces four additional Control Words, which are detailed in Section 5.4, *Framing Layer*, on page 26.

5.3.2 Burst Structure

5.3.2.1 Data Transmission Procedure

The bandwidth of the Interlaken interface is divided into data bursts from the supported channels. Data packets are transferred across the interface by means of one or more bursts, with the bursts delineated by means of one or more Control Words, as described in Section 5.3.2.2, *Control Word Format*, on page 15.

For the purpose of segmenting a packet of arbitrary size into bursts, the following two parameters are defined:

(i) **BurstMax**: The maximum size of a data burst (a multiple of 64 bytes)

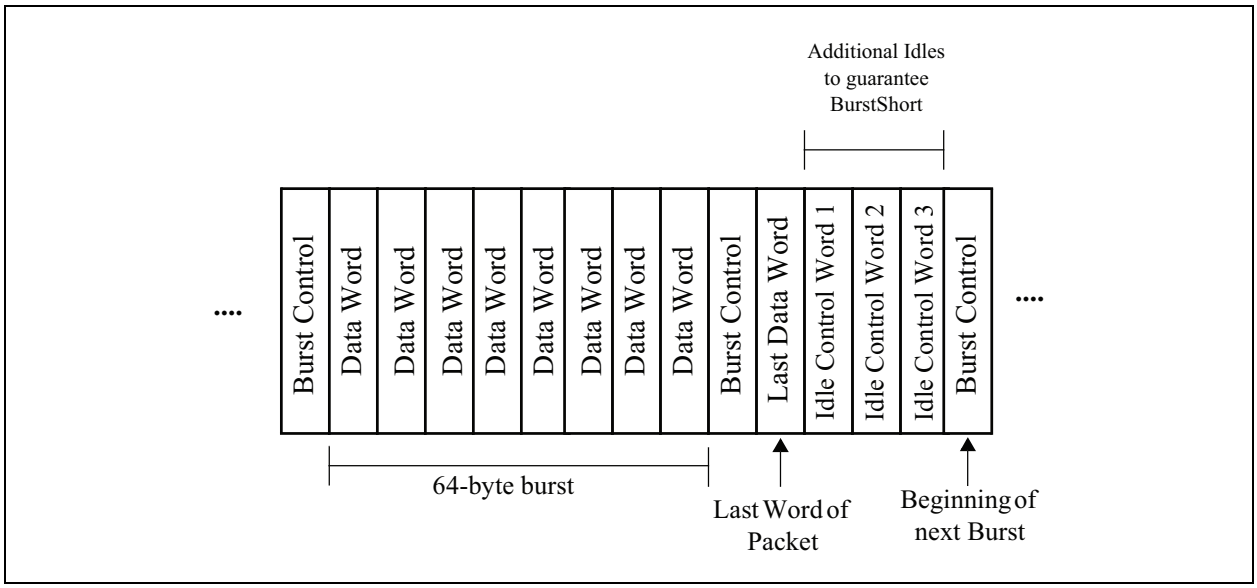
(ii) **BurstShort**: The minimum size of a data burst (a minimum of 32 bytes, with 8-byte increments)

The interface typically operates by sending a burst of data of **BurstMax** length, followed by a Control Word. The scheduling logic in the transmitting device is free to choose the order in which channels are serviced, subject to the constraint of the flow control state. Bursts are transmitted on each channel until the packet is completely transferred, at which point a new packet transfer on that channel may begin.

Because the interface is channelized, end-of-packet may occur back-to-back on several channels with a very small amount of remaining data on each channel. As both transmitter and receiver memories may be ideally designed with a wide datapath, they would need to be clocked at very high rates to handle this scenario. To reduce this burden on the receiver and transmitter, the **BurstShort** parameter guarantees a minimum separation between successive Burst Control Words. The minimum **BurstShort** interval is 32 bytes, with larger values possible in increments of 8 bytes.

Figure 6 illustrates the minimum separation guaranteed by **BurstShort**. **BurstShort** is enforced by adding extra Idle Control Words before the next Burst Control Word. In the example below, the EOP_Format for Idle Control Word 1 indicates EOP and the appropriate size for the Last Data Word, and the CRC24 of Idle Control Word 1 covers both the Last Data Word and Idle Control Word 1. Idle Control Word 2 and Idle Control Word 3 are inserted to maintain **BurstShort**, and the following Burst Control Word pertains to the data sent after it.

Figure 6 BurstShort Guarantee Illustration



5.3.2.1.1 Optional Scheduling Enhancement

The simple scheduling described above results in some unused bandwidth at the end of a packet for certain combinations of packet length and **BurstMax**. When the packet length modulus **BurstMax** is small, such that there is a small amount of data remaining to transfer after the last **BurstMax**, extra Idle Words are transmitted to enforce the **BurstShort** guarantee In the worst case, this unused bandwidth amounts to (**BurstShort** - 1) bytes per packet. However, by looking ahead in the packet to identify the location of the EOP, more efficient scheduling is possible. The following procedure illustrates one such mechanism, and is offered as an optional guideline for optimizing the performance of the interface.

This guideline introduces an additional parameter:

(iii) **BurstMin**: Defined to be a multiple of 32 bytes, subject to the constraints that **BurstMin** <= **BurstMax**/2 and **BurstMin** >= **BurstShort**; the usage of this parameter is defined below.

The following additional variables are defined for the purpose of this illustration:

- packet_length = the total length of the packet
- packet_remainder = the amount of data in the packet remaining to be sent once data transfer has begun
- data_transfer = the amount of data transferred on the current burst
- i = the number of bursts required to transfer the packet

The decision algorithm governing the burst size calculation is as follows:

The decision algorithm governing the burst size calculation is as follows:

```
packet_remainder = packet_length
for (x=1; x <= i; x++) {
if (packet_remainder >= BurstMax + BurstMin) then
    data_transfer = BurstMax
else
    if (packet_length MOD BurstMax < BurstMin) && (packet_remainder > BurstMax) then
        data_transfer = BurstMax - BurstMin
    else
        data_transfer = packet_remainder
packet_remainder = packet_remainder - data_transfer
}
```

This function guarantees that the last burst of a packet is of a size between **BurstMin** and **BurstMax**, avoiding the problem of multiple short end-of-packet segments. However, in order for this algorithm to operate properly, **BurstMin** cannot exceed half of **BurstMax**.

As an example, a packet of length 513 bytes is to be transferred across an Interlaken interface with **BurstMax** = 256 bytes and **BurstMin** = 64 bytes. In this case three bursts are sent:

- Burst 1 = BurstMax = 256 bytes
- Burst 2 = BurstMax - BurstMin = 256 - 64 = 192 bytes
- Burst 3 = packet_remainder = 65 bytes

If instead the packet was 511 bytes, only two bursts are sent:

- Burst 1 = BurstMax = 256 bytes
- Burst 2 = packet_remainder = 255 bytes

Implementations may tune the **BurstMax** and **BurstMin** parameters as desired, subject to the constraints defined above.

This optional algorithm is intended to guide implementations toward an efficient mechanism of transporting bursts. However, There is no additional burden placed on the receiving logic if the transmitter follows a different procedure for segmenting packets, as long as the **BurstShort** and **BurstMax** parameters are observed. As an example, there may be situations in converting from one interface type to another where reformatting bursts would impose an unnecessary burden. Other scheduling algorithms are possible, and designers are free to create them subject to the constraints defined above.

5.3.2.2 Control Word Format

Bursts are delineated by means of an 8-byte Control Word. The Control Word is identified in the data stream by using the ‘x10’ control code for bits[66:64] (defined in Section 5.4.2, *64B/67B Encoding*, on page 26) and bit[63] = ‘1’. The Burst and Idle Control Word formatting is illustrated in Figure 7 on page 16:

Figure 7 Control Word Format

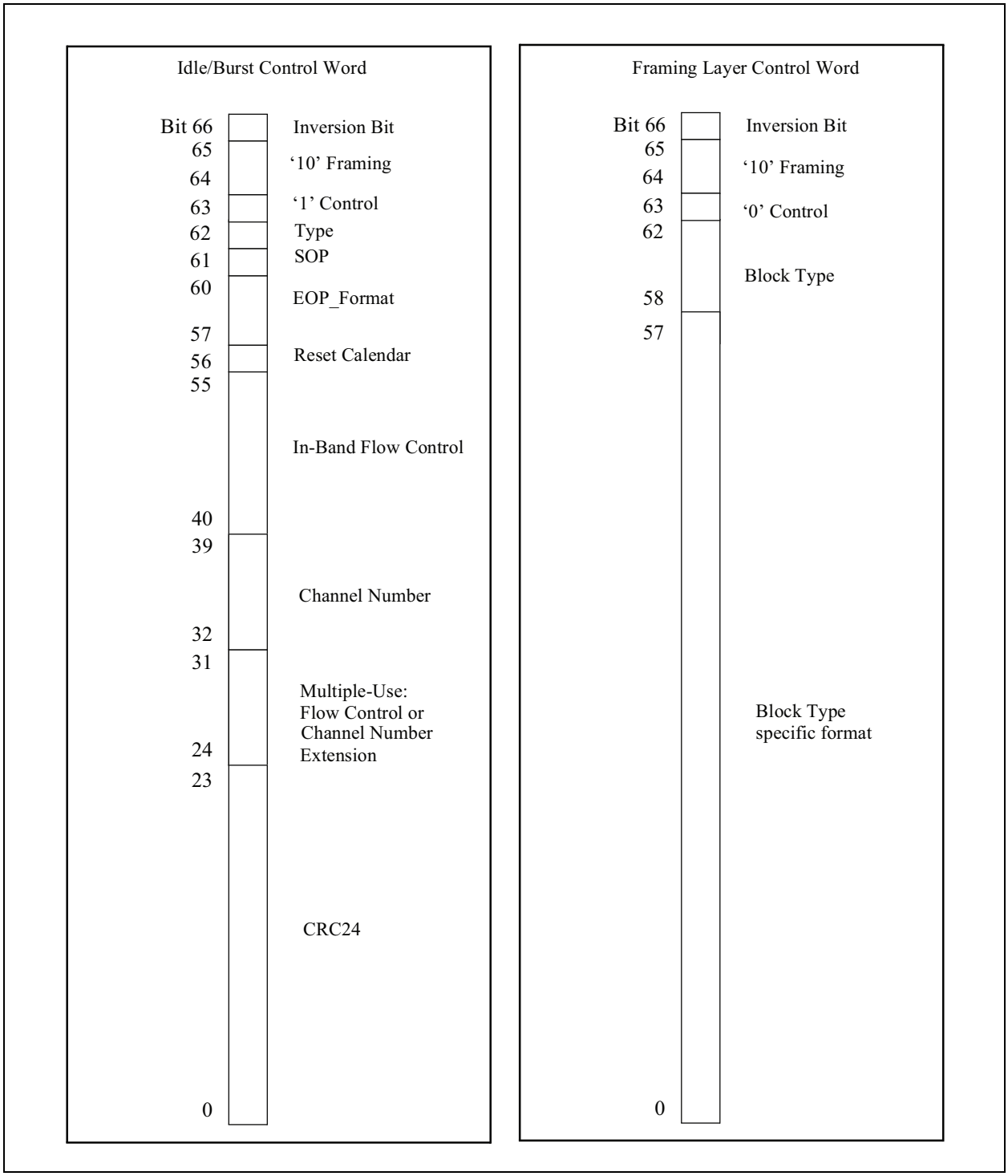


Table 1 Idle/Burst Control Word Format

Field	Bit Position	Function
Inversion	66	Used to indicate whether bits [63:0] have been inverted to limit the running disparity; 1 = inverted, 0 = not inverted
Framing	65:64	64B/67B mechanism to distinguish control and data words; a '01' indicates data, and a '10' indicates control
Control	63	If set to '1', this is an Idle or Burst Control Word; if '0', this is a Framing Layer Control Word (see Section 5.4, *Framing Layer*, on page 26)
Type	62	If set to a '1', the channel number and SOP fields are valid and a data burst follows this control word (a 'Burst Control Word'); if set to a '0', the channel number field and SOP fields are invalid and no data follows this control word (an 'Idle Control Word')
SOP	61	Start of Packet. If set to a '1', the data burst following this control word represents the start of a data packet; if set to a '0', a data burst that follows this control word is either the middle or end of a packet
EOP_Format	60:57	This field refers to the data burst preceding this control word. It is encoded as follows: '1xxx' - End-of-Packet, with bits[59:57] defining the number of valid bytes in the last 8-byte word in the burst. Bits[59:57] are encoded such that '000' means 8 bytes valid, '001' means 1 byte valid, etc., with '111' meaning 7 bytes valid; the valid bytes start with bit position [63:56] '0000' - no End-of-Packet, no ERR '0001' - Error and End-of-Packet All other combinations are left undefined.
Reset Calendar	56	If set to a '1', indicates that the in-band flow control status represents the beginning of the channel calendar
In-Band Flow Control	55:40	The 1-bit flow control status for the current 16 calendar entries; if set to a '1' the channel or channels represented by the calendar entry is XON, if set to a '0' the channel represented by the calendar entry is XOFF
Channel Number	39:32	The channel associated with the data burst following this control word; set to all zeroes for Idle Control Words
Multiple-Use	31:24	This field may serve multiple purposes, depending on the application. If additional channels beyond 256 are required, these 8 bits may be used as a Channel Number Extension, representing the 8 least significant bits of the Channel Number. If additional in-band flow control bits are desired, these bits may be used to represent the flow control status for the 8 calendar entries following the 16 calendar entries represented in bits[55:40]. These bits may also be reserved for application-specific purposes beyond the scope of this specification.
CRC24	23:0	A CRC error check that covers the previous data burst (if any) and this control word

Burst Control Words (Type = '1') identify the beginning of a data burst. Each burst data transfer must begin with a Burst Control Word, and this indicates that the SOP and Channel Number fields apply to the data immediately following. When the Burst Control Word falls between data bursts, the EOP_Format and CRC fields apply to the data immediately preceding, and the SOP and Channel Number fields apply to the data immediately following (the intention is to operate similarly to the SPI4.2 burst control semantic).

Idle Control Words (Type = '0') are always transmitted when there is no new data available to send. Because the flow control information must always be sent to the receiving device, the flow control fields are valid in both Idle and Burst Control Words, and the transmitter always sends valid flow control status in both types of control words.

The EOP_Format Field of the Burst Control Word identifies how many bytes of the last data word of the burst are valid. Bytes that are invalid are discarded by the receiver. By convention, the first valid byte occurs at bit field [63:56], the second valid byte at bit field [55:48], etc.

Data and control integrity is ensured by means of the 24-bit CRC. The CRC24 is calculated against all data in the burst and all the fields in the Control Word. The CRC24 polynomial is selected from [4]:

$$x^{24} + x^{21} + x^{20} + x^{17} + x^{15} + x^{11} + x^{9} + x^{8} + x^{6} + x^{5} + x + 1$$

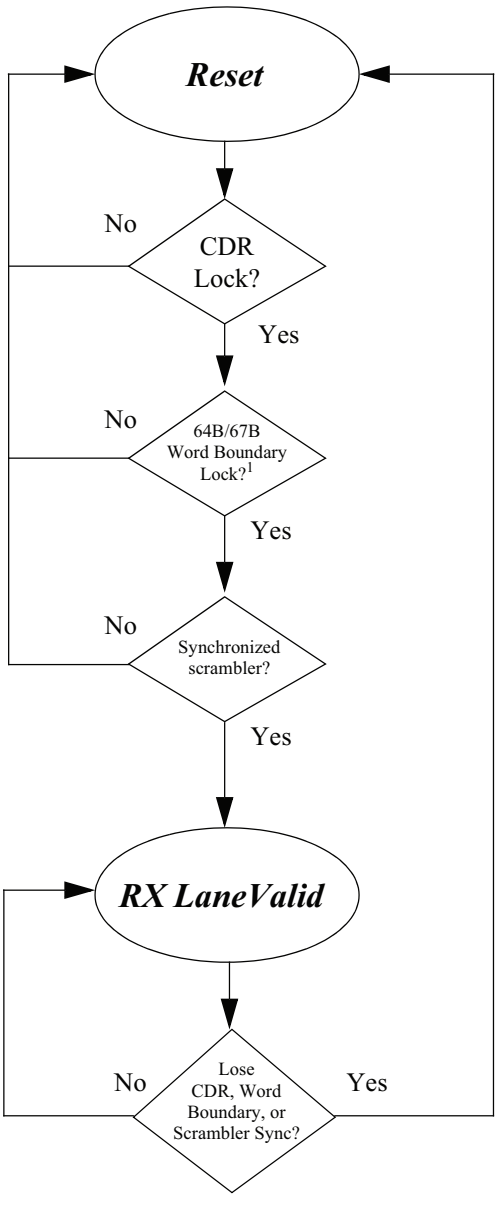
The details of the CRC computation are specified in Appendix B, *CRC and Scrambler Calculation Details* on page 48.

5.3.3 State Diagrams

The following state diagrams are provided to illustrate the logical operation of important components of the interface.

Each receive SerDes lane of the interface operates according to Figure 8.

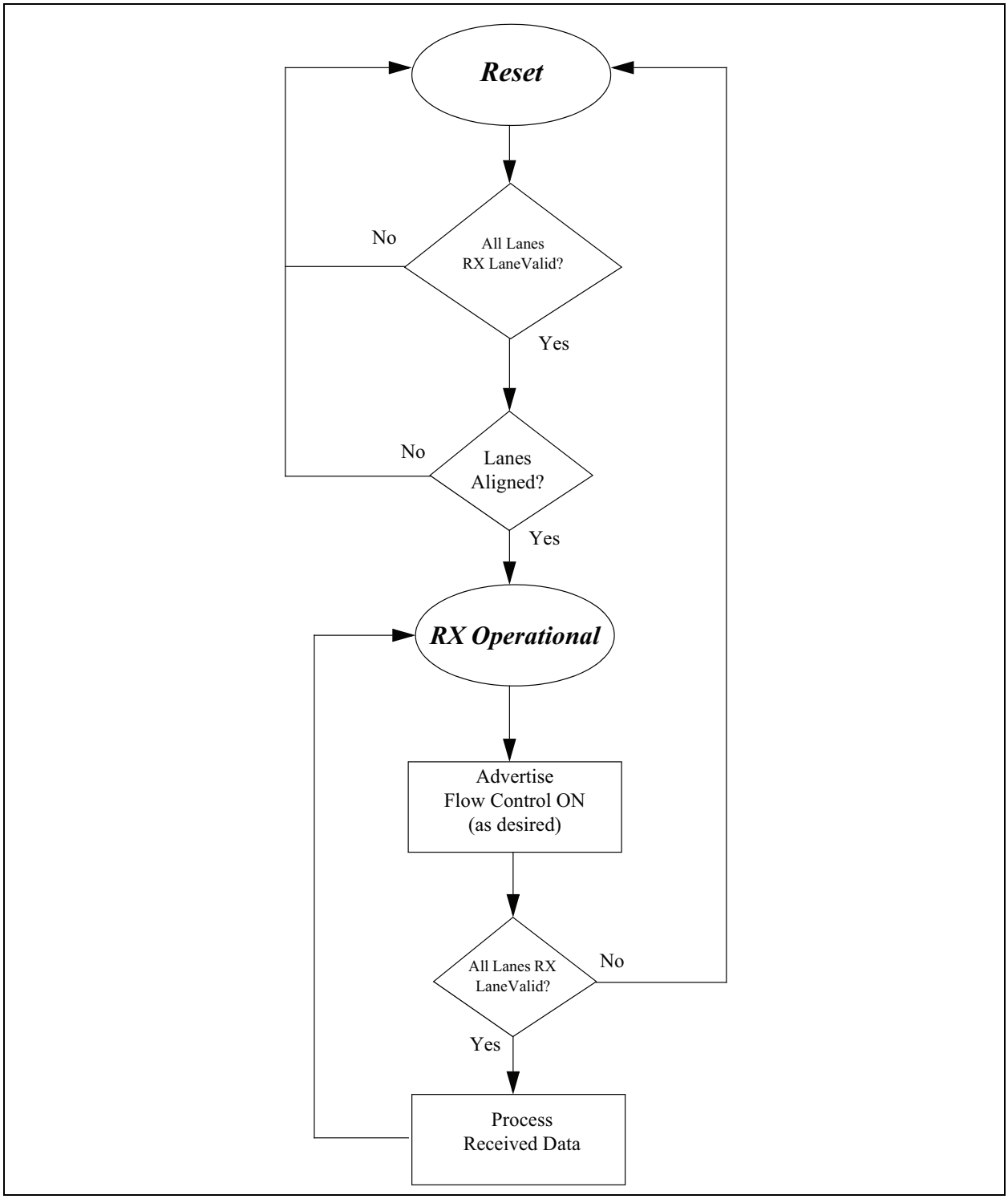
Figure 8 Receive Per-Lane State



Note:
1. The 64B/67B Word Boundary Lock state diagram is shown in Section 5.4.2, *64B/67B Encoding*, on page 26

The receive side of the interface (all lanes bundled together as a logical whole) then operates according to the following:

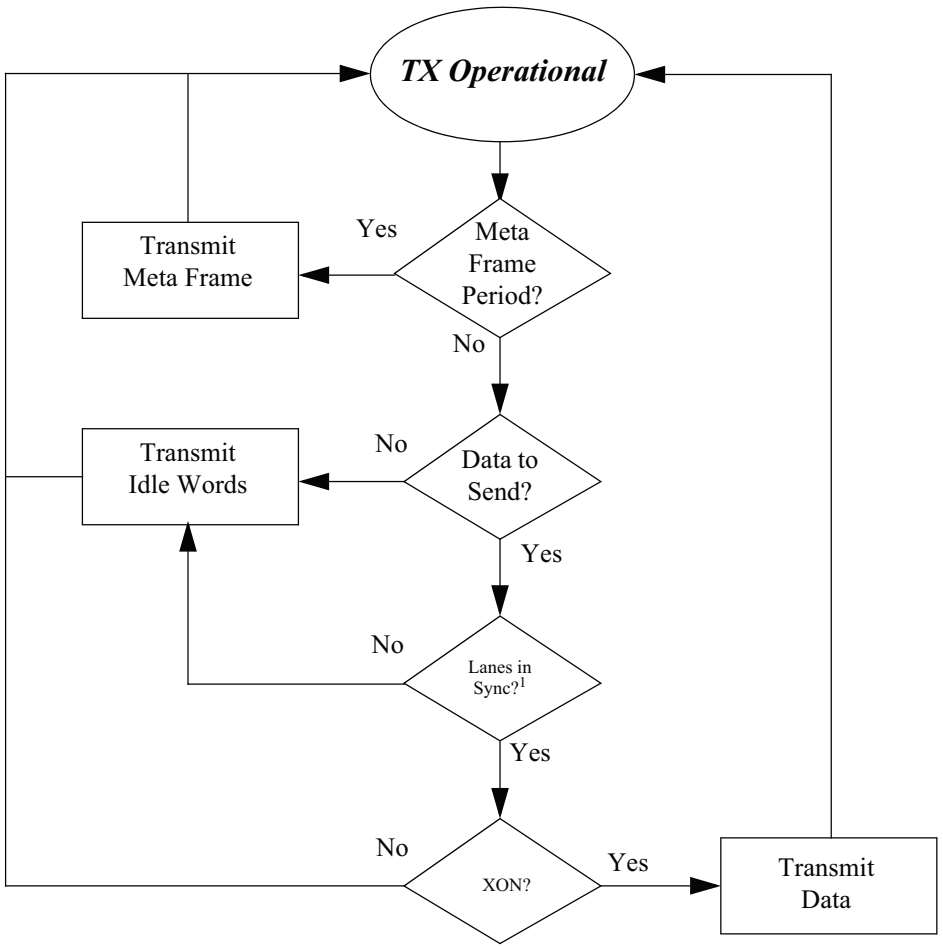
Figure 9 Receive Interface State



Once the receive side of the interface has entered the RX Operational state, the interface is free to advertise to the transmitter permission to send by signaling the ON state on all channels.

The transmit side of the interface (all lanes bundled together as a whole logical interface) operates according to the diagram in Figure 10:

Figure 10 Transmit Interface State



Note:
1. This optional test may be automatically performed in hardware, or controlled externally by software

5.3.4 Flow Control

A key feature of Interlaken is the ability to communicate per-channel backpressure. To provide this function, two options are specified: an out-of-band flow control interface and an in-band channel. Semantically, the flow control information uses a simple on-off mechanism to signal permission to transmit on a particular channel.

5.3.4.1 Protocol

The on-off flow control status is communicated with a single bit of status for each supported channel. By convention, a ‘1’ is chosen to identify the ‘XON’ state, indicating permission for the transmitter to send data on that channel. A ‘0’ identifies the ‘XOFF’ state, indicating that the transmitter should cease sending data on that channel.

There is no concept of credits with this protocol; once a channel is indicated as XON, the transmitter may send as much data as it chooses on that channel until the flow control status is changed to XOFF. The threshold whereby the receiver chooses to switch between the XON and XOFF states is a programmable option left to the user and is dependent upon the number of channels supported, depth of receive buffers, and the flow control latency of the given environment.

The flow control channels may optionally be mapped to a calendar, so that the flow control may be mapped to any set of calendar entries. By way of example, these could consist of a one-to-one mapping of channel to calendar entry, a one-to-many mapping to increase the frequency of certain channels, or the insertion of null fields to match devices with different channel definitions.

This calendar structure may also be used to provide link-level flow control, whereby a bit in the calendar represents the permission to transmit data on the interface as a whole. The polarity of the link status will be identical to that of the channel status: a ‘1’ indicates permission to transmit, while a ‘0’ indicates to cease transmitting immediately. To enable this function, each calendar entry can be configured either for channel information or link information. To facilitate low latency link status, the interface needs to provide enough calendar entries to program the link status in every Burst/Idle Control Word in the same bit position of those words. By way of example, and using greater than 16 channels, this could be performed by:

First Control Word:

Calendar Entry 0 = link status

Calendar Entry 1 = channel 0 status

Calendar Entry 2 = channel 1 status

...etc.

Second Control Word:

Calendar Entry 15 = link status

Calendar Entry 16 = channel 15 status

...etc.

Using this method, the link status would always appear in bit position [55] of the Burst / Idle Control Word.

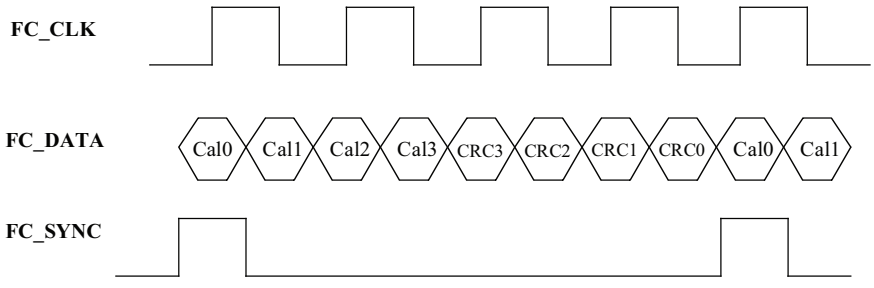
5.3.4.2 Out-of-Band Flow Control

To support systems that require simplex operation, an out-of-band flow control option is defined. This is implemented as a source-synchronous interface, and is specified with the following signals:

FC_CLK:	The clock to which the flow control data is synchronized
FC_DATA:	The flow control status information (single bit)
FC_SYNC:	A sync signal used to identify the beginning of the flow control calendar

The pad technology for each of these signals may be either LVDS or LVCMOS. The logical timing relationship of these signals is shown below:

Figure 11 Out-of-Band Logical Timing Diagram



An Example 4 calendar entry system

The out-of-band flow control channel is protected with a 4-bit CRC calculation that covers up to 64 bits of flow control data. Based upon the recommendations in [3], the CRC4 polynomial is:

$$x^4 + x + 1$$

The details of the CRC Computation are specified in Appendix B, *CRC and Scrambler Calculation Details* on page 48. When the number of channels is 64 or fewer, the CRC4 checksum occurs immediately following the last calendar slot, and is followed by the flow control status of calendar slot 0. When the number of calendar slots is greater than 64, the CRC4 checksum occurs once for every 64 bits of flow control status. For the last group of calendar slots, the CRC4 checksum occurs after the last supported calendar slot, followed immediately by the calendar slot 0 status.

As shown in Figure 11, FC_CLK is used to clock FC_DATA on both the rising and falling edges. At the maximum rate of 100 MHz, for a hypothetical implementation supporting 48 channels and 24 Gbps, the worst-case data in flight is:

FC_CLK_{period} =	10 ns
Time in flight =	(10 ns) / (2 bits/clk) * (48 channels + 4 CRC bits) = 260 ns
Data in flight =	(260 ns) * (24 Gbps) = 780 bytes

For an implementation supporting 256 channels and 24 Gbps, the worst-case data in flight is:

Time in flight =	(10 ns) / (2 bits/clk) * (256 channels + 16 CRC4 bits) = 1.36 µsec
Data in flight =	(1.36 µsec) * (24 Gbps) = 4.08 KB

5.3.4.2.1 Out-of-Band Flow Control Interface Timing

This section describes the AC timing parameters for the out of band flow control interface.

In order to provide the maximum capture window, the FC_DATA and FC_SYNC signals are sent at a data rate twice the clock frequency and are sent in quadrature phase with respect to the FC_CLK signal. Note: The timing relationship is the same for both the rising and falling edge of the clock.

Figure 12 Out-of-Band Flow Control Timing Diagram

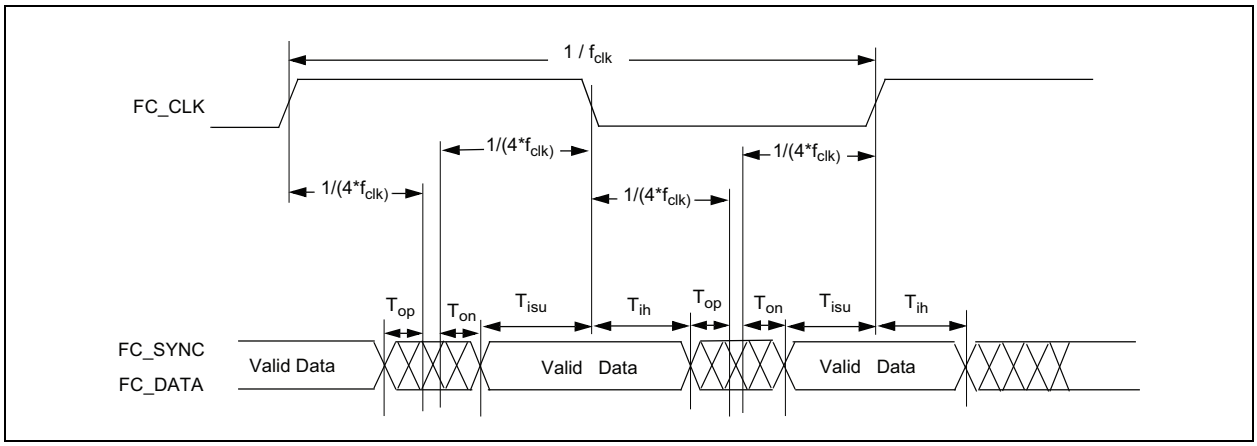


Table 2 Out-of-Band Flow Control Interface Timing

Symbol	Parameter	Min	Typ	Max	Units
f_{clk}	FC_CLK Clock Frequency	0		100	MHz
	FC_CLK Clock Duty Cycle	45		55	%
T_{isu}	Input Data/Sync setup time w.r.t clock edge.	0.75			ns
T_{ih}	Input Data/Sync hold time w.r.t clock edge.	0.75			ns
T_{on}	Next Output Data/Sync Invalid time w.r.t previous quadrature point of next clock edge.			0.75	ns
T_{op}	Previous Output Data/Sync Invalid time w.r.t. quadrature point of previous clock edge.			0.75	ns

5.3.4.3 In-Band Flow Control

When utilizing this option, the receiver makes use of flow control status transmitted in the Control Words sent across the interface as part of the normal data transfer. This option is provided for full-duplex implementations that require a minimum number of external signal pins.

As shown in Figure 7 on page 16, the Flow Control field of the Control Word is 16 bits, located in bit positions [55:40]. Bits [31:24] of the Control Word may also be used for 8 more bits of Flow Control, for a total of 24. These status bits represent the ON-OFF flow control status for each Interlaken calendar channel, with current calendar entry X at bit [55], calendar entry X+1 at bit [54], and so forth. To synchronize the start of the calendar the Reset Calendar bit is provided in the Idle/Burst Control Words; when this bit is a ‘1’, calendar entry 0 status appears in bit [55]. When Reset Calendar is a ‘0’, the calendar continues sequentially from where it left off in the previous Control Word. Once all the

channels' status has been communicated, the transmitter sets the Reset Calendar bit and the sequence repeats. Extra bits not required in the last Control Word of the calendar (i.e., when the number of channels is not a multiple of the number of status bits) are ignored by the receiver and set to 0 by the transmitter.

Because the Control Word CRC24 covers the Flow Control field, there is no requirement for an independent error check, and the CRC4 calculation performed for the Out-of-Band option is not preserved here.

Flow control information is always sent in both Idle and Burst Control Words.

Because Control Words are sent between each burst data transmission, the worst case frequency of flow control information is one message every maximum burst length. It is left to the implementer to select the BurstMax required for the desired flow control bandwidth.

As an example performance calculation, for an interface with a 256-byte burst and 48 channels, the data in flight during the calendar transmission is:

Data in flight = (2 bursts) * (256 bytes/burst) +
(2 control words) * (8 bytes/control word) = 528 Bytes

5.3.4.4 Full-Packet Mode Flow Control

While Interlaken is optimized to operate by interleaving transmissions from different channels, it also accommodates applications that require complete packet transmissions. For these applications the transmitting device simply avoids switching from one channel to another until the current channel's packet completes transmission.

There are two interpretations of flow control in full-packet mode: stop transmission immediately upon receipt of an XOFF message, or finish the current packet before stopping transmission. The first interpretation reduces the receiver buffering required before responding to flow control, at the expense of head-of-line blocking other channels in the interface; the second interpretation offers the opposite trade-off. Because different applications require different behaviors, this specification leaves open the possibility of choosing either or both interpretations in compliant implementations.

5.3.4.5 Flow Control Extension

Some applications may wish to implement a different flow control methodology than that provided by Interlaken's XON/XOFF; by example, this could involve the use of explicit credits exchanged by transmitter and receiver. Rather than attempt to specify this directly, Interlaken provides for this as a higher-layer function that may be implemented using an additional channel(s) to carry this information. By treating this extension as part of the data payload, any higher-layer protocol may be devised and reliably transported by the Interlaken interface.

5.4 Framing Layer

5.4.1 Overview

Interlaken defines a multifunction framing method to achieve simple and reliable transport, which consists of the following components:

Table 3 Overview of Framing Layer

Function	Purposes
64B/67B Encoding	Distinguish 8-byte word boundaries; Distinguish control and data words; Bound the baseline wander
Synchronous Scrambler	Guarantee bit transition density; Eliminate error multiplication
Lane Alignment	Align all the lanes within a bundle
Diagnostic	Provide diagnostics and optional per-lane status messaging
Skip	Compensate for clock differential in an electrical repeater
Rate Matching	Optimize receiver design by matching the data rate of the Interlaken with the data rate of the downstream services

Each of these are defined in the following sections. The framing layer uses Framing Layer Control Words as shown in Figure 7 on page 16. Bits [63:58] are used to distinguish this type of control word, with bit [63] being set to zero and bits [62:58] indicating the Block Type.

5.4.2 64B/67B Encoding

An encoding/scrambling method is required for a serial interface to delineate word boundaries, provide randomness to the EMI generated by the electrical transitions, allow for clock recovery, and maintain DC balance. The encoding protocol selected for Interlaken is a modification of the 64B/66B used for the IEEE 802.3ae 10 Gigabit Ethernet specification.

The existing 802.3 64B/66B solves the problem of word boundary delineation by combining a scrambled payload with two additional unscrambled bits prepended onto each 64-bit data or control word. If these sync bits are “01” they signify a data word, and if they are “10” they signify a control word; the combinations “00” and “11” are not allowed. By searching for the valid patterns in the received data stream, the receiving device declares word boundary lock after 64 correct matches, and it maintains lock by continually fixing on these two bits.

One weakness of this approach, however, is an unbounded baseline wander. Baseline wander, or DC imbalance, is caused by the accumulated excess of 1’s or 0’s transmitted on an individual SerDes lane. An electrical transition has an associated time constant, which in high-speed interfaces often does not allow a full voltage swing before the next bit is transmitted. Therefore, a sustained imbalance in either the number of 1’s or 0’s can produce a movement in the center voltage of the differential pair’s eye opening. Analysis of the 64B/66B scrambler polynomial shows that over a 64Kbit time scale a running disparity in excess of +/- 1,000 bits can occur, which can produce excessive eye shifts, cause complications in the design of receiver circuitry, and increase the bit-error rate.

To bound this effect, Interlaken inverts the sense of the bits in each transmitted word such that the running disparity always stays within a +/- 96-bit bound. Each lane of the bundle maintains a running count of the disparity: a ‘1’ bit increments the disparity by one, and a ‘0’ bit decrements the disparity by one. Before transmission, the disparity of the new word is calculated and then compared to the current running disparity. If the new word and the

existing disparity both have the same sign, the bits within the new word are inverted. A framing bit is supplied in bit position 66 so the receiver may identify whether the bits for that word are inverted, as below:

Table 4 Inversion Bit 66

Bit 66	Interpretation
0	Bits [63:0] are not inverted; the receiver may process this word without modification
1	Bits [63:0] are inverted; the receiver must un-invert before processing this word

This code is referred to in this document as 64B/67B. All bits in every word, including bit 66, are included in the running disparity count. When bit 66 is set to a '1', bits [63:0] are inverted. The legal values of the three sync bits are:

Table 5 Sync Bits Encoding

Bits [66:64]	Interpretation
001	Data Word, no inversion
010	Control Word, no inversion
101	Data Word, bits [63:0] are inverted
110	Control Word, bits [63:0] are inverted
All others	Illegal states

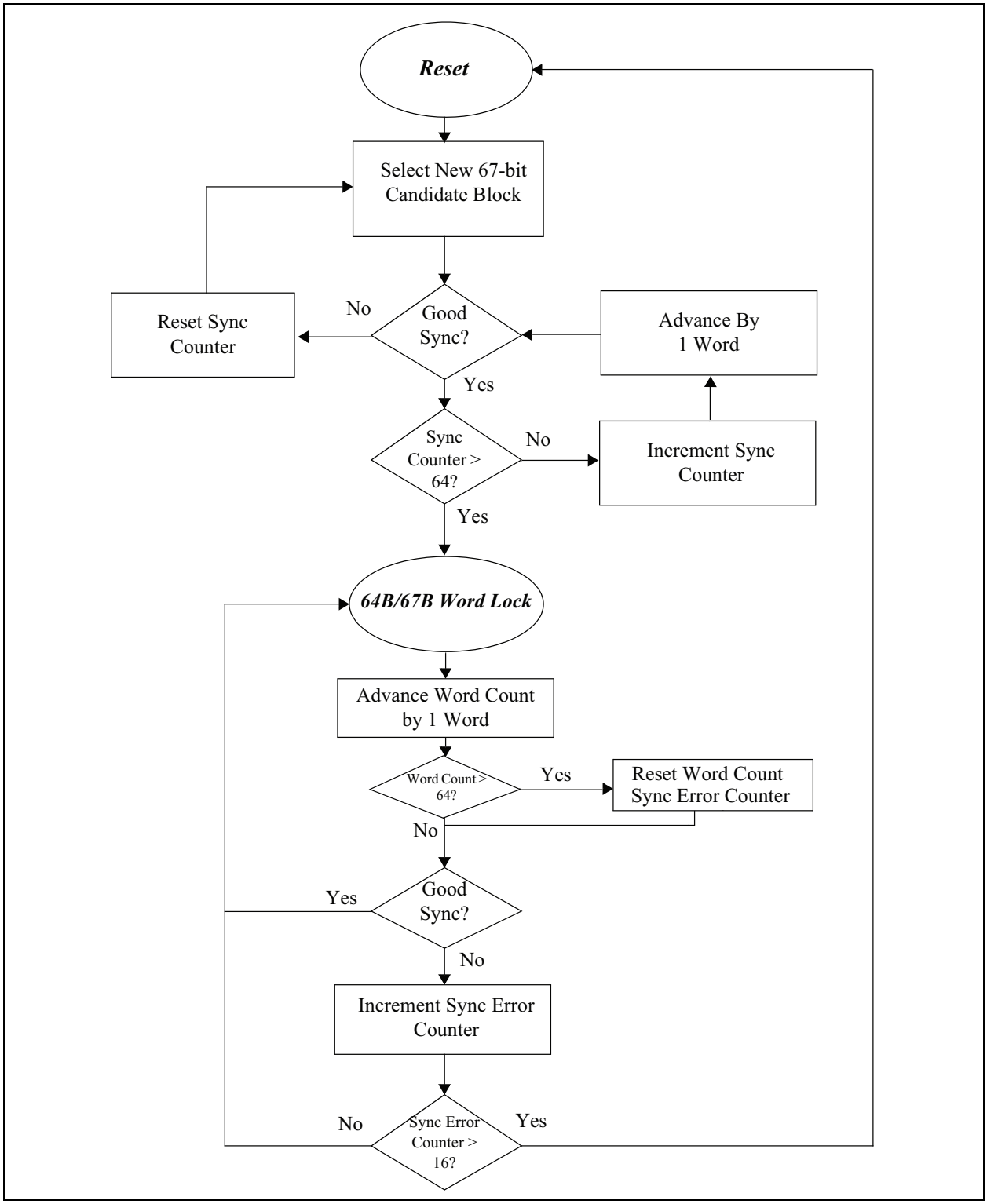
The IEEE's 64B/66B code defines a procedure for locking to the sync bits. The receiver searches for a transition from high to low or low to high (the only legal sync codes), and selects this as a hypothetical sync pattern. In the next framing bit position, the receiver again looks for one of the legal patterns; if a legal pattern occurs again it repeats this procedure, and if it does not it resets its state and searches for another legal pattern. In order to declare lock the receiver must observe 64 consecutive legal sync patterns.

With the 64B/67B code, Interlaken adds an additional sync bit, but only 50% of the possible combinations of these three bits are legal, the same as 64B/66B. As such, to achieve lock with an identically low probability of an incorrect sync, 64 consecutive legal sync patterns (defined in Table) must be observed by the receiver.

The 64B/67B encoding creates an overhead of 4.5%.

The flow diagram for achieving and maintaining 64B/67B word boundary lock is shown in Figure 13 on page 28:

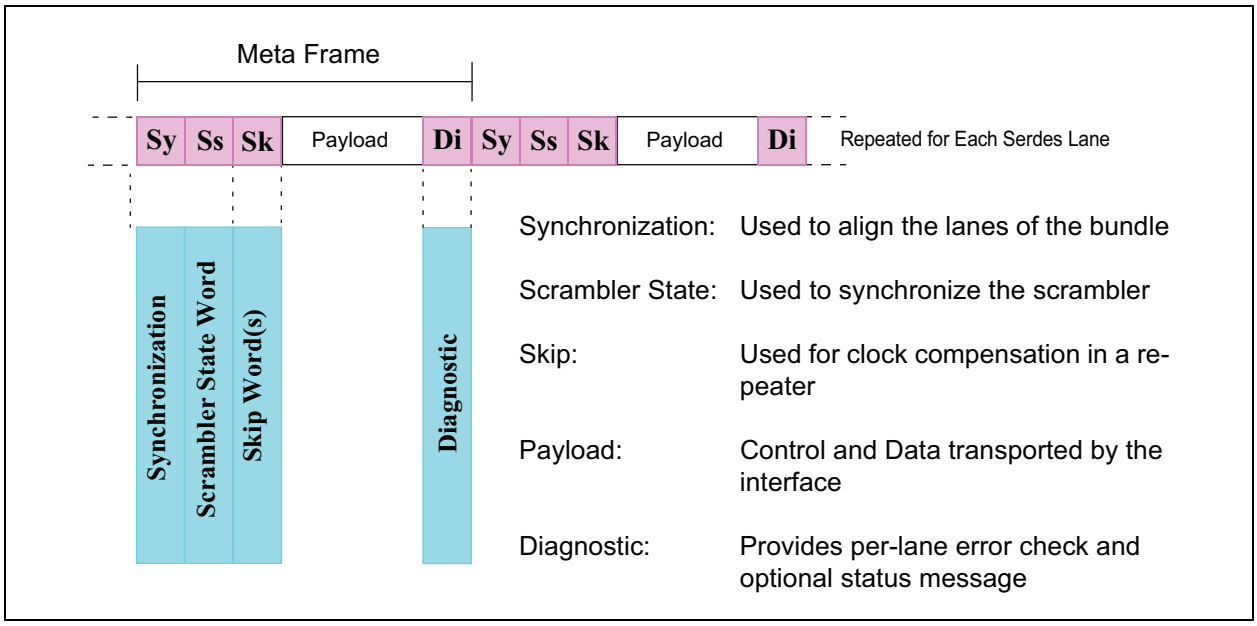
Figure 13 64B/67B Word Boundary Lock



5.4.3 Meta Frame

The Interlaken framing method introduces the concept of a Meta Frame. The Meta Frame is defined as the per-lane set of the Synchronization, Scrambler State, Skip, and Diagnostic words, along with the payload data (burst data and control information) carried on each lane. Figure 14 illustrates the structure:

Figure 14 Meta Frame Structure (Per Lane)



The size of the Meta Frame is a single programmable parameter, MetaFrameLength, that applies to all lanes of the bundle. It represents the sum of the data payload and one set of Synchronization, Scrambler State, Skip, and Diagnostic words. The Meta Frame structure is orthogonal to the data transmissions; these Meta Frame control words may occur at any point within a data burst.

In addition to Synchronization, Scrambler State, and Diagnostic Words, a Skip Word is defined to provide clock compensation for electrical repeater applications. For reference, the unique Block Types of each Meta Frame Control Word are shown in Table 6:

Table 6 Meta Frame Control Word Block Types

Meta Frame Control Word	Block Type (positive disparity)	Block Type (negative disparity)
Synchronization	011110	100001
Scrambler State	001010	110101
Skip	000111	111000
Diagnostic	011001	100110
Note: Figure 15, Figure 18, Figure 19, Figure 21, and Figure 26 all show Meta Frame Control Words using positive disparity Block Types for clarity. The negative disparity versions of these Control Words are also valid.		

The details of the Synchronization, Scrambler State, Diagnostic, and Skip Words are described in the following sections.

5.4.4 Synchronous Scrambler

The 802.3 64B/66B code uses a self-synchronous scrambler on the payload. This has the advantage of not requiring any synchronization; the scrambler state is a function of the received data stream and can be recovered after the length of the scrambler (58 bits) are received. But this scrambler uses two feedback taps, and as such it has the property of replicating errors twice, so that a single-bit error on the line becomes three single-bit errors at the receiver. Because Interlaken stripes data across the lanes within a bundle, this multiplication can push bit errors across words. The next errored word may or may not be part of the same burst, which means that the location of errors is no longer restricted within the burst. For multiple-bit errors this can reduce the error detection properties of the CRC24 and is an undesirable artifact.

To eliminate this scenario Interlaken employs an independent synchronous scrambler on each lane of the interface. The synchronous scrambler does not feed the input data back upon itself; rather each bit is XOR'd with the current state of the scrambler, so no error multiplication may occur. The scrambler polynomial is:

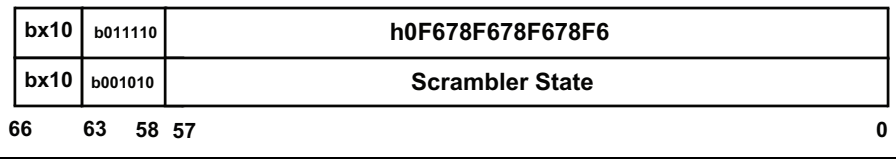
$$x^{58} + x^{39} + 1$$

The scrambler polynomial is activated after device reset, and the transmitter never resets it again. Instead, the current scrambler state is sent to the receiver to allow it to decode the data which follows. The scrambler advances and rolls over indefinitely during interface operation, with the exception that it does not advance during the transmission of the unscrambled Synchronization and Scrambler State Words.

There is no requirement that each lane use the same scrambler state, and to minimize cross-talk between lanes, implementations should initially reset the scrambler to different values on each lane; the only restriction is that the scrambler never be reset to all zeroes. Because the scrambler state is explicitly forwarded in the datapath, there is no need for the receive side of the interface to know to what value the transmit scrambler was reset.

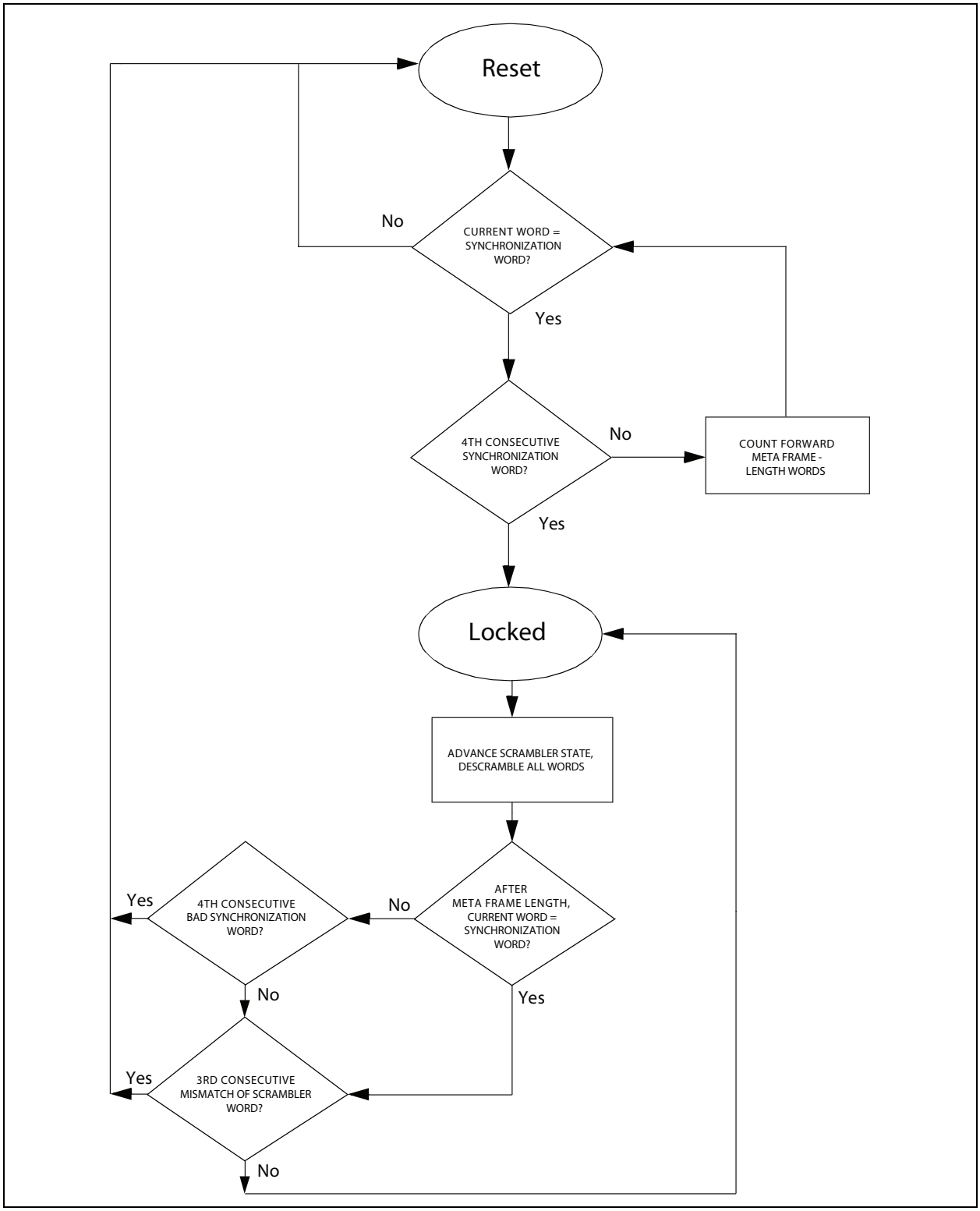
In order to correctly decode the received data, the receiver must be synchronized with the state of the scrambler polynomial. Interlaken synchronizes via the combination of a unique 64-bit Synchronization Word and a Scrambler State Word that are transmitted consecutively as part of the Meta Frame:

Figure 15 Synchronization and Scrambler State Words



To allow for synchronization at the start of operation and after errors, the Synchronization and Scrambler State Words are transmitted unscrambled. Within the reset state, each lane searches for the unique pattern of the Synchronization Word. If the received word is the Synchronization Word (matches all 64 bits), the receiver counts until a MetaFrameLength (measured in 8-byte words) quantity of data has passed and test for another Synchronization Word. If it identifies the Synchronization Word it begins the sequence again, until it has identified four consecutive Synchronization Words. The state flow is shown in Figure 16 on page 31:

Figure 16 Scrambler Synchronization State Diagram



Once synchronization is achieved, the interface uses the recovered value of the scrambler polynomial from the Scrambler State Word to seed the descrambler. All data and control words, with the exception of Synchronization and Scrambler State Words, are scrambled from bits [63:0]; framing bits [66:64] are never scrambled. Each lane should verify that the scrambler state received in each Scrambler State Word after synchronization is consistent with its current expected scrambler state, and if not, signal an error after three consecutive mismatches as specified in Section 5.4.11.3, *Bad Scrambler State*, on page 39.

The size of the Meta Frame is always exactly **MetaFrameLength**. Because Interlaken provides for the addition or removal of a Skip Word to manage clock compensation in an electrical repeater, the repeater may need to adjust the position of the Synchronization Word relative to how it was originally transmitted (see Section 5.4.7, *Clock Compensation*, on page 33). This always occurs, however, such that a receiver observes a constant quantity of data between Synchronization Words.

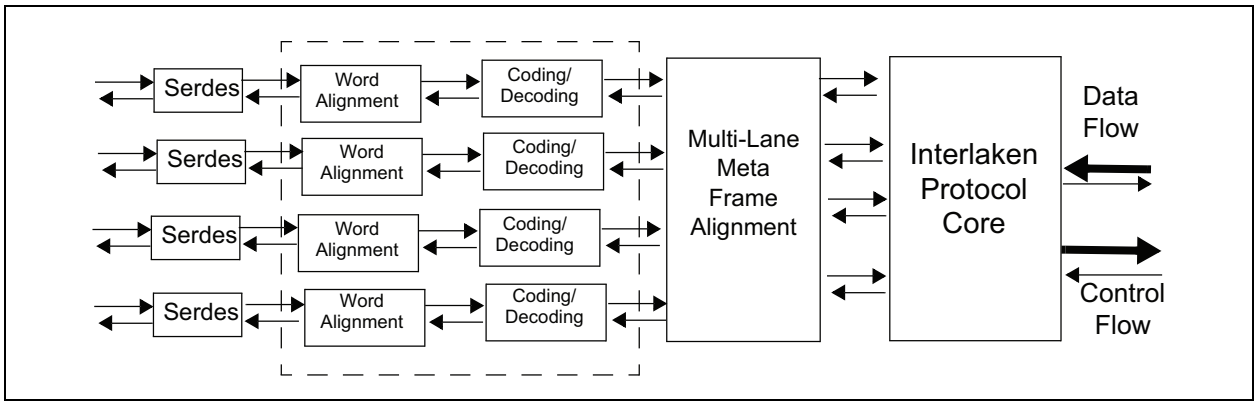
If the Synchronization Word is not identified, the receiver signals that an error has occurred. If four consecutive Synchronization Words are unidentified, the receiver returns to the Reset state and begins to search for the Synchronization Word. If three consecutive Scrambler State values contradict the receiver's expected scrambler state (all on the same lane), the receiver declares an error and attempts to resynchronize the scrambler.

5.4.5 Lane Alignment

Once the word boundaries are identified and the scrambler properly reset, the lanes of the bundle must be aligned. Interlaken guarantees that Synchronization Words are sent across the interface at a fixed frequency to regularly align the datapath SerDes lanes. To achieve alignment, the Synchronization Word is transmitted simultaneously across all lanes. The receiver then identifies these words, measures the skew between them across the lanes of the bundle, and adjusts its internal skew compensation logic accordingly. The architecture of this logic is left as an implementation choice; Interlaken only defines the means by which alignment may be achieved.

The transmission frequency of Synchronization Words is defined by the **MetaFrameLength**.

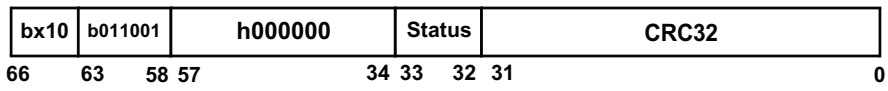
Figure 17 Interlaken Lane Alignment Segmentation (4-Lane Example)



5.4.6 Lane Diagnostics

The Diagnostic Word is identified with the Block Type value of **0b011001**. The format of the Diagnostic Word is shown in Figure 18:

Figure 18 Diagnostic Word

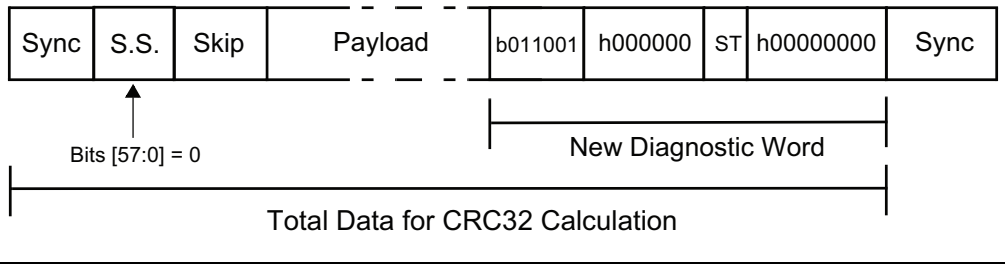


There are two functions assigned to the Diagnostic Word - a lane Status Message and per-lane error detection. The 2-bit Status field defines a place for a per-lane status message that is sent from receiver to transmitter, and its function is defined in Appendix A. The CRC32 is provided as a diagnostic tool on a per-lane basis, so that errors on the interface may be traced to an individual lane. It is calculated over all the words transmitted within the Meta Frame, before scrambling and inversion, except for the 64B/67B framing bits, but including bits [63:0] of the Diagnostic Word itself, with the CRC32 field padded to all zeros. For ease of implementation, the 58-bit scrambler state within the Scrambler State Word is also treated as all zeroes when computing the CRC32. The CRC32 polynomial is taken from [4]:

$x^{32}+x^{28}+x^{27}+x^{26}+x^{25}+x^{23}+x^{22}+x^{20}+x^{19}+x^{18}+x^{14}+x^{13}+x^{11}+x^{10}+x^{9}+x^{8}+x^{6}+1$

For ease of calculation, the fields over which the CRC32 are calculated are shown in Figure 19:

Figure 19 CRC32 Calculation Illustration



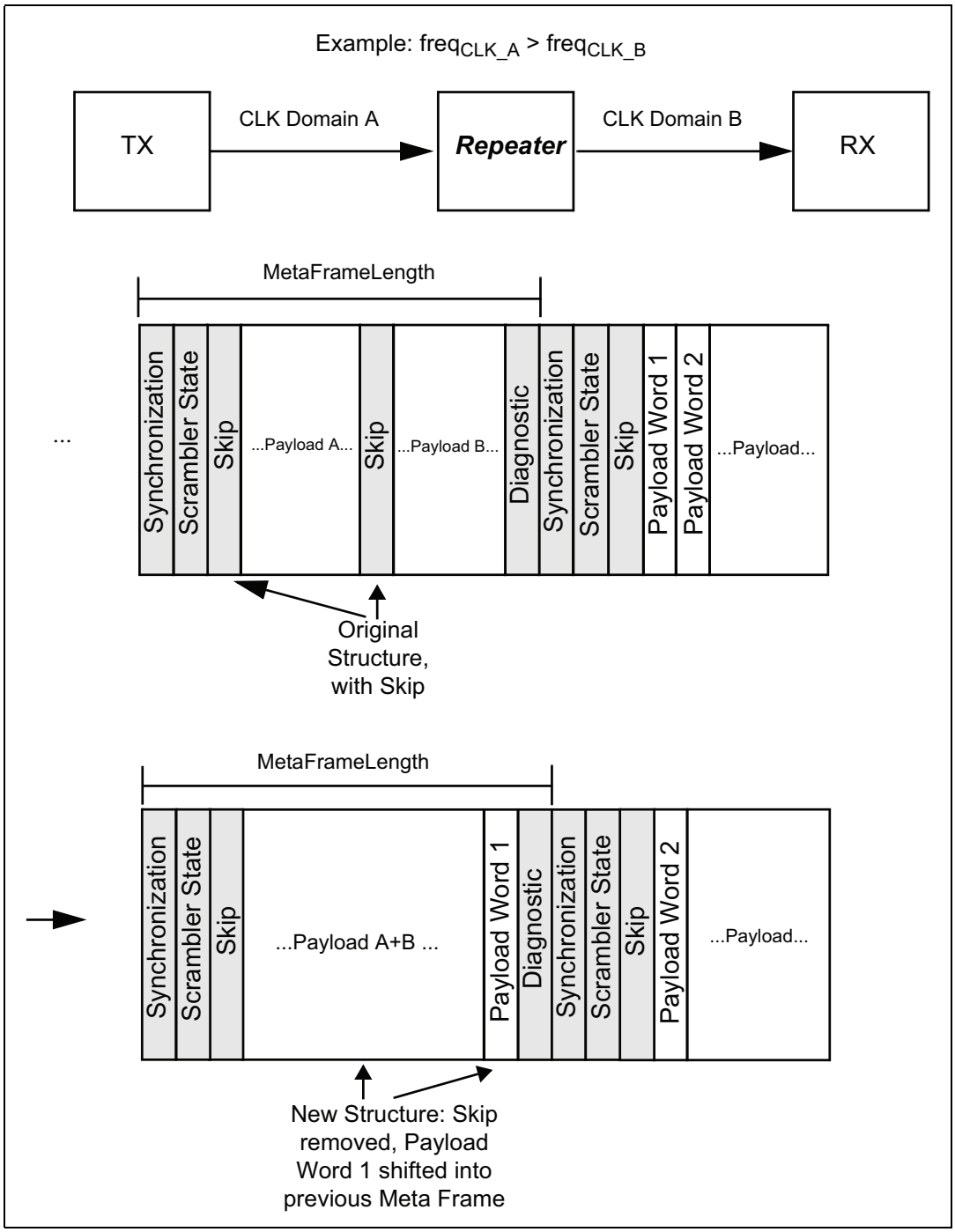
Diagnostic Words are counted as part of the **MetaFrameLength** just as Synchronization, Scrambler State, Skip, Data, and Burst/Idle Control Words.

5.4.7 Clock Compensation

The purpose of a Skip Word is to enable clock compensation for a repeater function, by which the protocol may be electrically relayed across an intermediary device. There can be a slight difference in clock rate on each side of the repeater, and to bridge this gap it is necessary to periodically remove a Skip Word if the second clock is slower than the first, or to add a Skip Word if the second clock is faster than the first. A single Skip Word is defined as a required part of the Meta Frame, but additional Skip Words may be added at any point in the Meta Frame, except between the Diagnostic, Synchronization, and Scrambler State Words. It is mandatory for receivers to correctly identify and remove them from the received data.

If there is a repeater between the original transmitter and ultimate receiver, the repeater may compensate for a slower transmit clock by silently discarding this Skip Word. If this occurs, the repeater must maintain the constant separation of MetaFrameLength between Synchronization Words. It performs this by shifting the first payload word of the next Meta Frame into the current Meta Frame, and scrambling it with the correct scrambler state at the end of the current Meta Frame. Figure 20 illustrates this procedure:

Figure 20 Clock Compensation Procedure



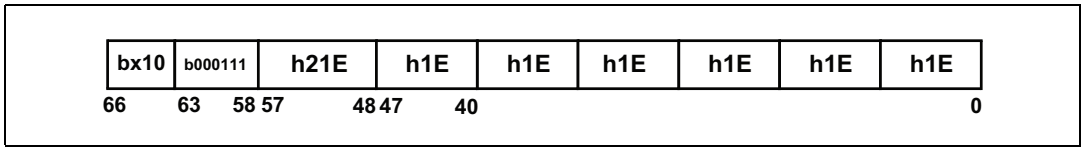
Note that this procedure of deleting Skip Words and shifting payload words from one Meta Frame into the previous Meta Frame eventually requires that the Diagnostic, Scrambler State, and Synchronization Words from one Meta Frame be shifted into the prior Meta Frame. To maintain consistent Meta Frame formatting, in this case the Diagnostic, Synchronization, and Scrambler State Words must be deleted. If a Diagnostic Word is deleted, the repeater should transmit a low-pass filtered version of the status message to avoid eliminating any transitory status information. The details of this process are left to the implementer.

If CLK_B in the above example is instead faster than CLK_A, the opposite approach is required - an additional Skip Word is added, and the last Payload Word of the current Meta Frame is shifted into the next Meta Frame. Eventually this process requires that a Diagnostic Word be added; in this case the status message should retain the same information as the immediate prior status message.

If the repeater determines that it needs to discard a word due to a clock difference on only a subset of all the lanes, it shall still discard all the words across the interface simultaneously, not just on the affected lane(s). Using a **MetaFrameLength** of 2K words, at most sixteen bytes is sent every 16KB, or at a ratio of 1:1,024. A 100ppm differential in clock frequency represents a ratio of 1:10,000, so this Meta Frame frequency meets this compensation requirement. Note that the **MetaFrameLength** may also be set shorter to enable quicker lane alignment or a smaller quantity of data over which the diagnostic CRC is calculated.

The Skip Word is identified by a Block Type value of **0b000111**. The format of the Skip Word is as follows:

Figure 21 Skip Word Format



5.4.8 Overhead

Because the Synchronization, Diagnostic, and Scrambler State Words are sent so infrequently they consume a minimal amount of interface bandwidth. The overhead is dependent on the size of **MetaFrameLength**, but for a hypothetical 2K words the worst-case overhead (with one Skip Word) is:

32/(16,384) = 0.20%

5.4.9 Skew Budget

Interlaken is specified to tolerate a worst-case skew between the individual lanes of a multi-lane interface of 107 UI (unit intervals), determined according to Table 7.

Table 7 Skew Budget

Skew Source	Budget (UI)
PMA Tx	67
PCB and Medium	40
Total	107

These budgets were derived according to the following observations:

PMA Tx: The SerDes driving each datapath lane are not required to share a transmit PLL, therefore the output of the transmitter could be skewed by the difference between two or more blocks of SerDes using different PLLs. It is expected that a conservative, worst-case implementation would use a 67-bit wide interface into each SerDes lane; if so, the maximum skew between two blocks of SerDes using different PLLs would be 67 bits, or 67 UI on the serial lane.

PCB & Medium: The application environment for Interlaken is expected to be similar to that defined for XAUI in 802.3ae-2002, and as such this parameter should scale with SerDes rate. As the highest performance SerDes used for Interlaken in the medium term is expected to be 6.375 Gbps, or twice the XAUI frequency, then this requirement is double the 20 UI requirement for XAUI, or 40 UI.

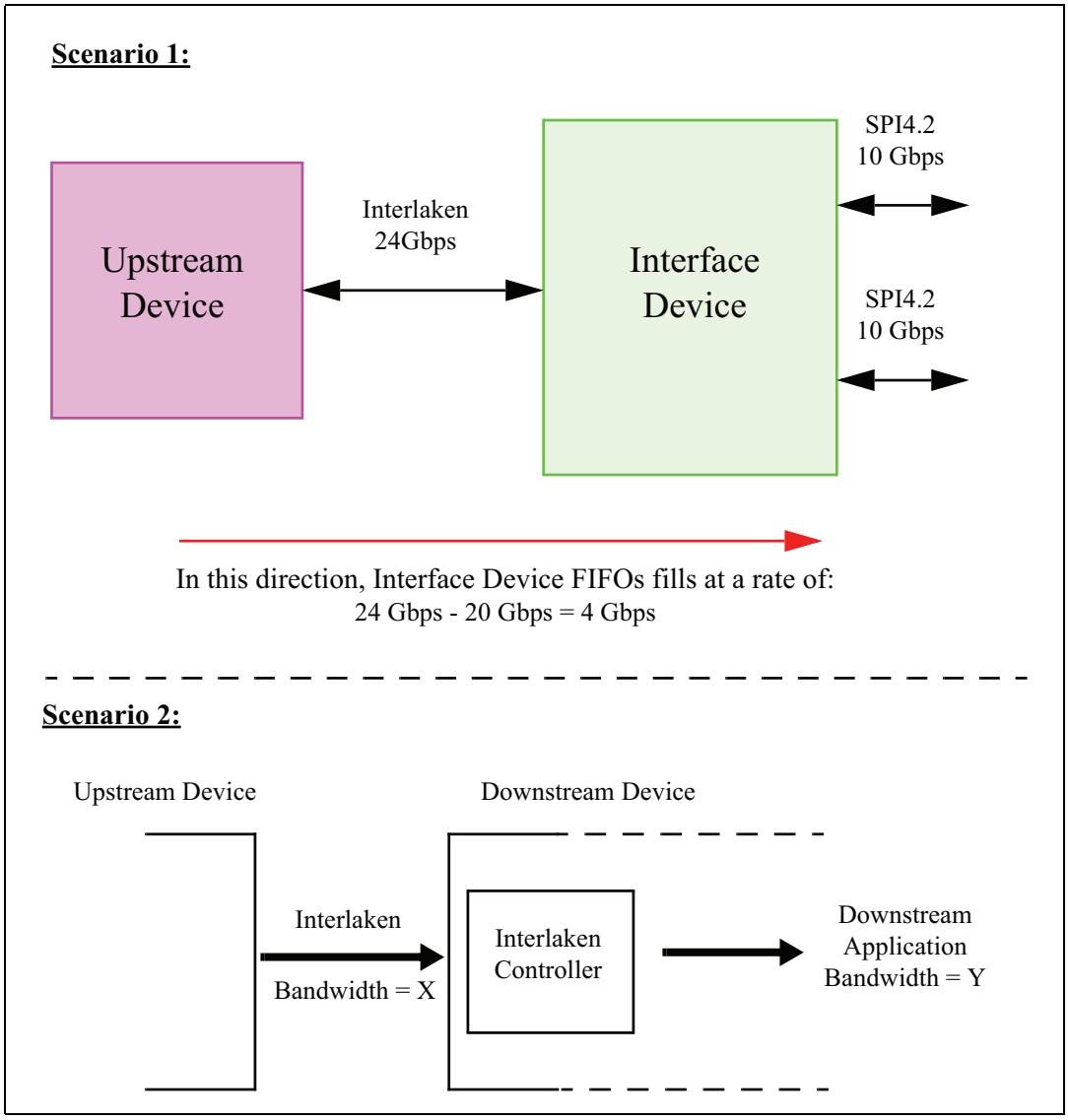
The receive PMA also creates skew that the Interlaken controller must compensate for, but as it is not necessary to specify this value to ensure interoperability, it is left to each implementation to account and correct for this skew.

5.4.10 Rate Matching

Some applications may wish to translate between Interlaken and an existing protocol such as SPI4.2. For these applications the bandwidth of the two interfaces may not match, creating a potentially expensive buffering function in the bridging device.

Additionally, there are situations in which the receiver buffering capacity may be reduced if the data rate can be guaranteed to be less than the maximum achievable rate. To provide for this optimization, Interlaken defines a mandatory rate matching function.

Figure 22 Rate Matching Scenarios



Interlaken provides rate matching by offering the ability to insert Idle Control Words into the datapath at a defined frequency to limit the bandwidth of data transferred across the interface.

The rate matching logic controls the throughput of the interface as a whole, rather than individual channels. The implementation shall be in the form of a single token bucket, which increments at the desired rate and decrements when data is transmitted. If the token bucket is empty the transmitter sends Idle Words until positive tokens are available. Idle Words must be sent between data bursts, so the rate matching logic shall have a worst-case latency of BurstMax before it can act. The granularity of the token bucket is be a programmable value, with a minimum granularity of one byte, such that it can match the granularity of the internal datapaths of the two devices using the interface.

Two parameters are introduced to define the rate matching function as shown in Table 8:

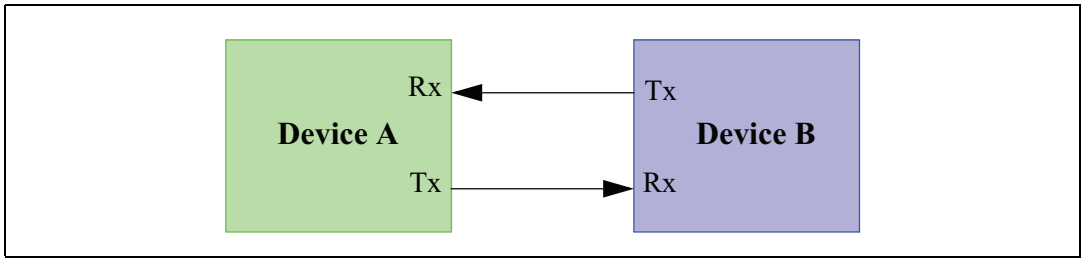
Table 8 Rate Matching Parameters

Parameter	Function
RateLimit	The overall rate which bounds the interface (bytes/sec)
BurstLimit	The maximum quantity of data that may be burst across the interface before invoking the rate limiting logic (bytes)

5.4.11 Error Conditions

Error conditions may apply to the interface as a whole as well as to individual bursts. Interlaken contains extensive error detection logic to provide a significant enhancement to the protection provided by existing protocols such as SPI4.2. Because different applications often require a different error handling responses, it is beyond the nature of this specification to mandate precisely how each error case should be handled. The following sections do, however, identify all likely error types and suggest possible ways that they may be handled. Please refer to Figure 23 for the specific references used below:

Figure 23 Device Reference



5.4.11.1 The Receive SerDes Loses Lock

If one of Device A's datapath SerDes loses lock on the recovered receive clock, Device A is no longer able to correctly receive data or burst control information from Device B. Because the burst data is striped across the set of lanes, random portions of each burst communication are lost when one lane is broken. In this case Device A should error open packets on all channels. Device A signals to Device B to stop transmitting by sending XOFF on all of its flow control status channels. Device B transmits Idle Control Words, as normal, once it receives this flow control status. Device A may also optionally send a Sync Message indicating the failure as defined in Appendix A, *Status Messaging* on page 46.

Device A immediately attempts to reacquire SerDes lock. Once it does so it then needs to reacquire word boundary alignment. The amount of data lost is dependent on clock recovery time and the total interface bandwidth.

Once Device A regains lock on its SerDes and correctly identifies 64 word boundaries, it signals its readiness to resume operation by advertising XON on its flow control channels. Device B resumes transmitting data as normal.

5.4.11.2 The Receive Logic Loses Word Boundary Sync

The receiving SerDes can lose word boundary sync when it fails to identify legal 3-bit patterns on bits [66:64] of each received word. The receiver declares itself out of lock only after it fails to find legal sync patterns on 16 words within a 64-word window, and declares that it is back in lock only after receiving a legal sync patterns on 64 consecutive words. Please refer to Figure 13 on page 28 for the state transitions of the word boundary sync algorithm.

Because this case is a subset of Section 5.4.11.1, *The Receive SerDes Loses Lock*, on page 38, the behavior of the interface is identical to that described above, with the exception that no time is required for the SerDes to reacquire lock.

5.4.11.3 Bad Scrambler State

It is possible that after initial synchronization, the received scrambler state may not match the expected current scrambler state. However, once the scrambler has synchronized it should never become unsynchronized, so this error should only occur in the presence of bit errors within the Scrambler State Word or alignment problems on the interface. Only after three consecutive scrambler state mismatches should the receiver declare an error and attempt to re-synchronize the scrambler.

5.4.11.4 Lane Alignment Fails

An interface is considered out of alignment if the datapath SerDes logic cannot find the Synchronization Word of the Meta Frame within the specified interval (107 UI) simultaneously on all datapath lanes. If the interface falls out of alignment, there is no way to reliably identify the correct data sequence. In this case the receiver should error all open packets and attempt to re-align on the next available Meta Frame.

To prevent a single bad Synchronization Word from disrupting alignment, the receiver should not declare loss of alignment until four consecutive alignments fail. To reacquire lost alignment, the receiver should also require four consecutive successful alignments.

5.4.11.5 Burst CRC24 Errors

Errors are detected by means of a mismatch in the Burst/Idle Control Word CRC. The CRC24 covers all data in the previous burst and bits [63:24] of the Burst/Idle Control Word. For ease of calculation, the last word of the packet, which contains invalid bytes if the packet is not a multiple of eight bytes long, has these invalid bytes set to all zeroes, and these bytes are also part of the CRC calculation. For the same reason the CRC24 field of the Control Word is also treated as if it contained all zeroes and is included in the CRC24 computation.

A CRC error indicates a corruption either within the current data or the control information. Because the Burst/Idle Control Word contains the channel number field, it is impossible to distinguish the channel associated with the following data burst; therefore all open channels should be errored if a CRC failure is detected.

5.4.11.6 Flow Control Errors

Because of the high frequency of flow control information, the only pathology associated with missing a message is a delay in communicating the flow control status. If an error is detected, the receiver should behave conservatively and assume that all channels are in the XOFF state until the next calendar reset and subsequent error-free status messages.

5.4.11.7 Unknown Control Word Types

If the interface receives a Control Word that it cannot interpret (e.g., it doesn't have either Control = '1' or one of the defined Block Type values) or is in the wrong position, it should be considered an error. This could occur when the framing bits are corrupted to a '10', if the interface loses word boundary alignment, or if the Block Type is corrupted. These control words should be discarded, and because they may be a Control Word that suffered an error, the conservative response is to error all open channels.

5.4.11.8 Bad 64B/67B Codewords

If the interface receives a 64B/67B codeword with one of the illegal framing patterns, it should discard it and error all open channels. This can only occur if either a data word or control word had their framing bits corrupted.

5.4.11.9 Diagnostic CRC32 Errors

The CRC32 is provided primarily as a diagnostic tool to allow errors to be traced to specific lanes and assist in quick fault detection. Additionally, an interface could use the identification of a CRC32 error as a real-time indication of link fault and remove that lane from service. The procedures necessary to achieve this are beyond the scope of this specification.

5.4.12 Lane Resiliency

Because the Interlaken protocol is independent of the number of lanes, resiliency may be provided by continuing operation in the presence of a failure on a single lane of a multi-lane implementation. The choice to continue operation in the presence of a single-lane failure is left as an optional feature, and is not required for compliance to the protocol. It is assumed that software intervention is required to reconfigure the interface to operate under these conditions. The Status Message feature of Appendix A may assist in providing this function.

5.5 Electrical Specifications

Interlaken is specified as a multi-lane full-duplex interface, using differential pairs connected to SerDes circuits on each end. Because the 8-byte block-coded words are striped across the individual lanes, there is no requirement on how many lanes to implement; the protocol scales from one to any number of lanes that are practical to allocate on a single IC. The protocol throughput scales with the rate used on each SerDes lane. It does not restrict which rate or electrical specification to use. The Interlaken Alliance has published interoperability guidelines which includes specific electrical and rate recommendations.

5.6 Recommended Statistics

The following interface statistics in Table 9 are defined as a recommendation only; it is not a requirement that they be implemented to claim compliance to this specification.

Table 9 Statistics (Sheet 1 of 2)

Statistic	Function
RX_Packets	Number of packets received (per channel)
RX_Bytes	Number of bytes received (per channel)
TX_Packets	Number of packets transmitted (per channel)
TX_Bytes	Number of bytes transmitted (per channel)

Table 9 Statistics (Sheet 2 of 2)

Statistic	Function
RX_Bad_Packets	Number of packets that are errored (i.e. bad CRC, FIFO overflow, ERR bit set, etc.; per channel)
RX_FIFO_Overflow	Number of packets dropped due to receive FIFO overflow (per channel)
RX_CRC_Error	Number of bursts with a detected CRC error
RX_FC_Error	Number of errors detected on the out-of-band flow control interface
RX_BurstMax_Error	Number of bursts received longer than the BurstMax parameter
RX_Alignment_Error	Number of alignment sequences received in error (i.e., those that violate the current alignment)
RX_Alignment_Failure	Number of times alignment was lost (after four consecutive RX_alignment_errors)
RX_Word_Sync_Error	Number of times a lane lost word boundary synchronization (per lane)
RX_CDR_Error	Number of times a lane lost clock-data-recovery (per lane)
RX_Lane_CRC_Error	Number of errors in the lane CRC (per lane)
RX_Bad_Control_Error	Number of words received with Control Word framing (‘x10’) that don’t match one of the defined Control Words

5.7 Test Patterns

The Interlaken controller must support test pattern generation and reception. The specific test patterns are modeled on those defined in 802.3ae-2002. They consist of two types: a programmable pattern and a PRBS31/23/7 pattern generator and checker.

The programmable pattern generator must be capable of storing a set of patterns and repetition values. The patterns should be transmitted as: PatternA * RepetitionA times, followed by PatternB * RepetitionB times, and so forth. The Interlaken controller should support a minimum of two programmable patterns, with the pattern length defined by SerDes requirements, and a minimum 8-bit repetition register per pattern. A programmable pattern check may also optionally be provided, dependent the particular SerDes test requirements.

Examples of programmable patterns are:

- High-frequency: 1010_1010_1010_1010_...
- Low-frequency: 1111_1111_0000_0000_...
- Mixed-frequency: 1111_1111_0101_0101_0000_0000_1010_1010_...
- Complex: a mix of high-density transitions, low-density transitions, and phase jumps

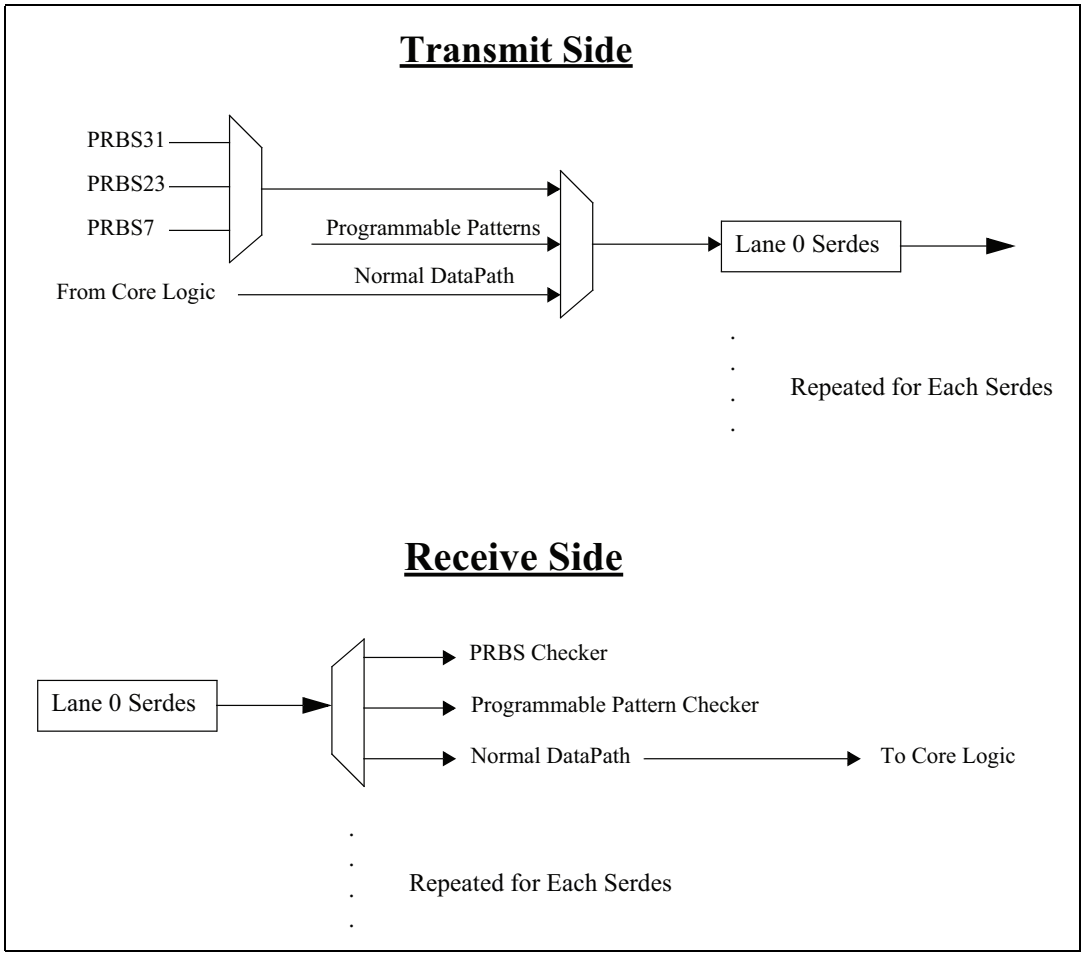
The PRBS pattern generator polynomials supported are shown in Table 10:

Table 10 PRBS Polynomials

Name	Polynomial
PRBS31	$x^{31} + x^{28} + 1$
PRBS23	$x^{23} + x^{18} + 1$
PRBS7	$x^{7} + x^{6} + 1$

The test pattern circuitry should be modeled along the high-level architecture shown in Figure 24:

Figure 24 Test Pattern Architecture



5.8 Latency Considerations

The latency of flow control response must be estimated in order to size the receive logic buffers. The following parameters are offered as a guideline to assist making this estimation; the values are specific to each implementation.

For the purposes of this illustration, the following buffer abstractions are defined:

Figure 25 Latency Illustration

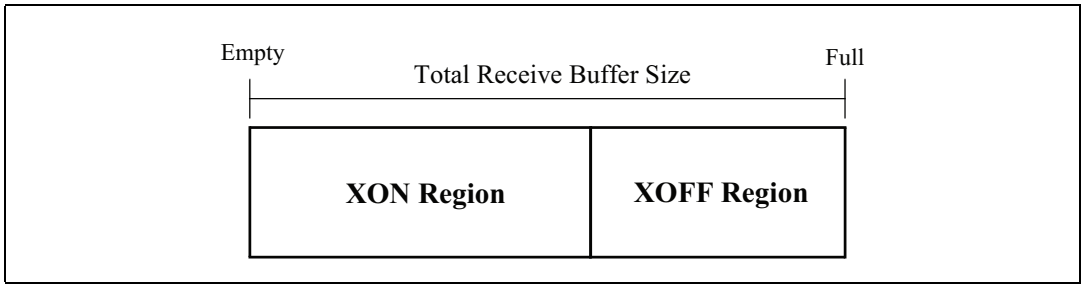


Table 11 Latency Parameters

Parameter	Symbol	Description
Status Change Latency	t*SCL*	Latency to create new status message after detecting a change from the XON to the XOFF Regions, or vice-versa
Status Transmit Latency	t*STL*	Latency to transmit the status information from receiver to transmitter
Transmitter Control Latency	t*TCL*	Latency of the transmitter to process the new status information
Transmitter Pipeline Latency	t*TPL*	Latency due to data already in the transmitter processing pipeline (transitioning XON -> XOFF) or latency to push new data through the pipeline (transitioning XOFF -> XON)

The turn-on and turn-off times are a function of these four steps:

1. The time for the receiver to detect a change between regions of the receiver buffer and generate new status
2. The time for the new status to be transmitted from the receiver to the transmitter
3. The time for the transmitter to process the new status and adjust its scheduling
4. The time to allow for data already in flight in the transmitter processing pipeline

The total flow control latency that must be considered is the sum of all these components:

$$t_L = t_{SCL} + t_{STL} + t_{TCL} + t_{TPL}$$

The size of the receive buffer required is therefore a function of the data rate of the interface, the flow control bandwidth, the size of the internal pipeline structure, and this latency time.

5.9 Performance

The performance of an interface may be understood in terms of the percentage of the raw bandwidth that is available for carrying a traffic payload. For Interlaken, this performance is shown in Table 12 for common traffic types.

Table 12 Efficiency Analysis

Lane Configuration	POS, 41-byte Frames	Ethernet, 65-byte Frames	Ethernet, 9601-byte Frames
Efficiency Factor	69.8	77.5	92.3

where:

Efficiency Factor = (Encoding Efficiency) * (Framing Efficiency) * (Alignment Efficiency) * (Meta Frame Efficiency) * 100%

Encoding Efficiency: The 95.5% efficiency of using 64B/67B encoding

Framing Efficiency: The impact of the 8-byte control word overhead as a percentage of the frame or cell size (256-byte **BurstMax** for this example)

Alignment Efficiency: The impact of invalid characters inserted to pad the end of a frame to an 8-byte word boundary

Meta Frame Efficiency: The 99.8% efficiency created by the Synchronization, Scrambler State, Diagnostic, and Skip Words (assuming a **MetaFrameLength** of 2K words, and not counting optional insertion of Idle Control Words for rate matching)

6.0 Bibliography

1. IEEE: 802.3ae-2002 - Amendment: Media Access Control (MAC) Parameters, Physical Layers, and Management Parameters for 10Gbps Operation.
2. Optical Internetworking Forum (OIF): OIF-SPI4-02.1 - System Packet Interface Level 4 (SPI-4) Phase 2 Revision 1: OC-192 System Interface for Physical and Link Layer Devices.
3. P. Koopman and T. Chakravarty, Cyclic Redundancy Code (CRC) Polynomial Selection for Embedded Networks, The International Conference on Dependable Networks and Systems, DSN-2004.
4. G. Castagnoli, S. Brauer, and M. Herrmann, “Optimization of Cyclic Redundancy-Check Codes with 24 and 32 Parity Bits”, IEEE Transactions on Communications, Vol. 41, No. 6, June 1993.

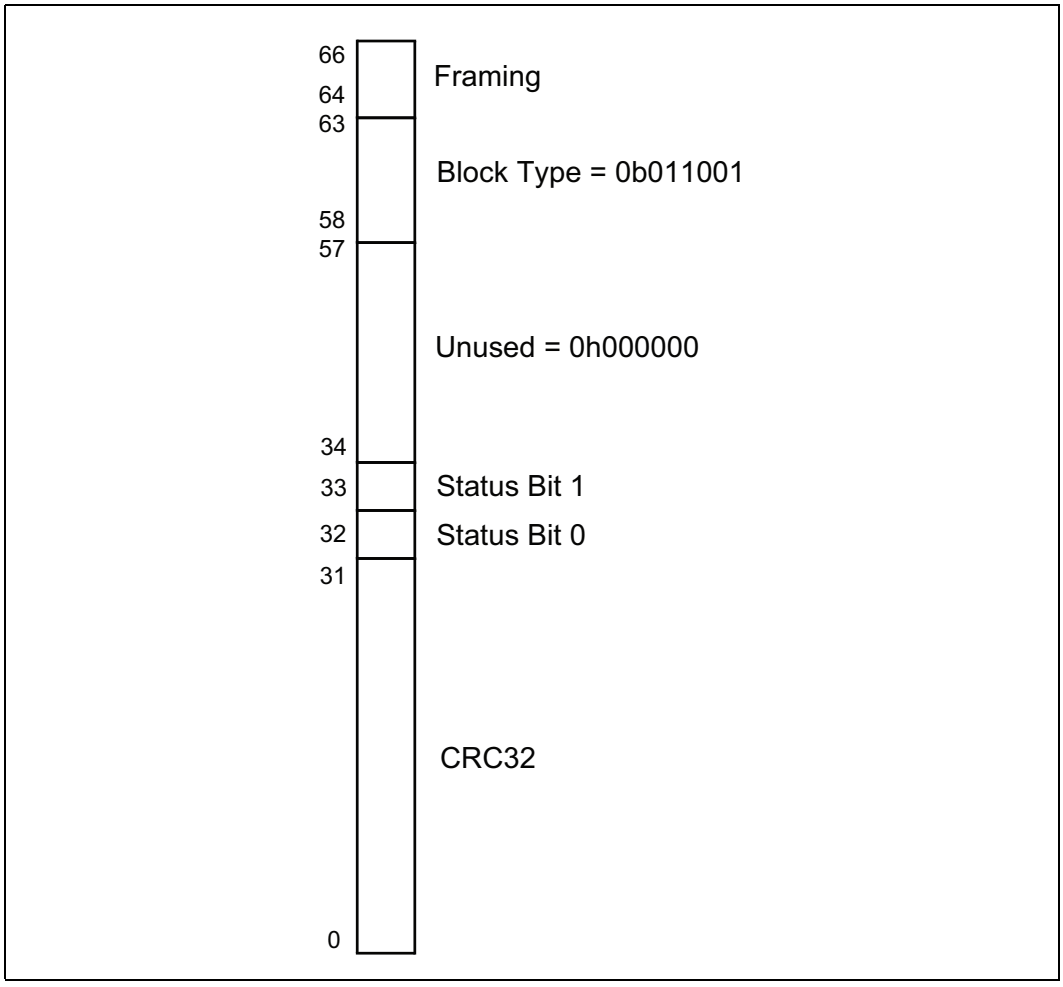
Appendix A Status Messaging

Some applications may desire that the receive side of an Interlaken interface be able to signal to the transmitter that one or more of its receive links are inoperable. This may serve the purpose of increasing the Alignment frequency to speedup the process of re-acquiring alignment, assist in quickly enabling a failover to redundant links, or improving the speed of alternate failover mechanisms. For this purpose the Status Message is defined as an optional extension to the Interlaken protocol.

Bi-Directional Interfaces

For bi-directional implementations, the Status Message is carried in bits [33:32] of the Diagnostic Word. The format of the message consists of a Status Bit 1 representing the health of this lane, and Status Bit 0 representing the health of the entire interface. A ‘1’ is defined to mean a healthy condition, and a ‘0’ to indicate a problem. The message is formatted as shown in Figure 26:

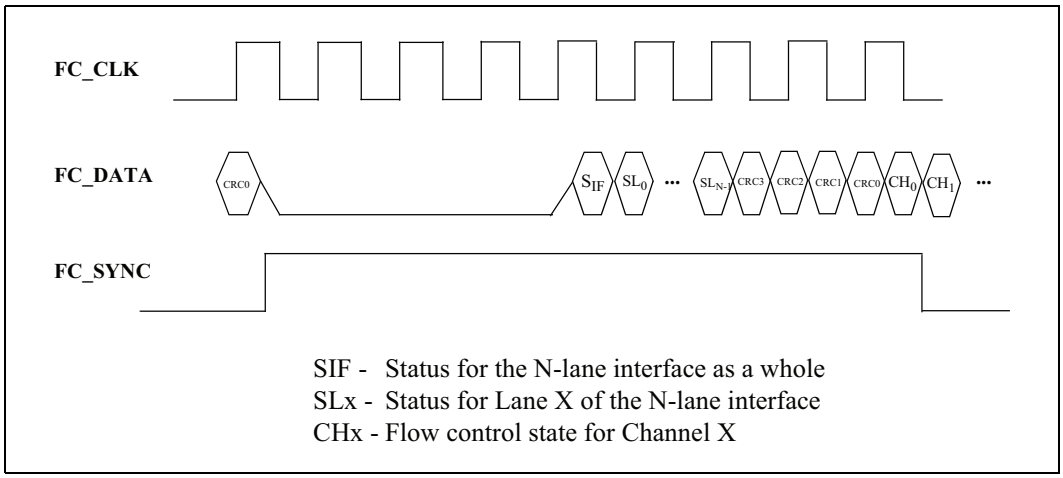
Figure 26 Status Message Format



Uni-Directional Interfaces

For uni-directional environments, the out-of-band status channel is used to communicate the status. In this case a modification to the out-of-band signalling protocol is defined as illustrated in Figure 27:

Figure 27 Out-of-Band Status Message



To avoid allocating flow control bandwidth to a Status Message that normally does not indicate any problems, the message is defined to appear only when one of the lanes identifies a problem. To prevent errors on the FC_SYNC line from inadvertently indicating a Status Message, the FC_SYNC signal is held high for eight contiguous bits before transmitting the Status Message, as well as for the duration of the Status Message. The Status Message consists of a bit (SIF) to indicate the health of the interface as a whole, plus a single bit per lane of the interface, encoded as described above; the message is as long as the number of lanes in the interface, plus one, plus the 4-bit CRC. The CRC4 function that protects the out-of- band status also protects the Status Message, and it is sent immediately after the last Status Message bit. It is only calculated to cover the Status Message, and operates orthogonally to the out-of-band status CRC4.

The transmission sequence is as follows:

- After detecting the lane problem, the receiver waits until it has finished transmitting the current Flow Control calendar;
- Next it holds the FC_SYNC line high for eight bits, then transmits the Status Message;
- After transmitting the last bit of the Status Message, the FC_SYNC line is held high for the first bit of the new Flow Control calendar, then driven low for the second bit of the calendar, and the Flow Control calendar resumes normal transmission;
- After the Flow Control calendar is transmitted in full, the Status Message repeats.

The Status Message alternates with the Flow Control calendar until the fault condition that initiated the Status Message is resolved.

Appendix B CRC and Scrambler Calculation Details

CRC:

Different bit ordering conventions are possible when implementing CRC functions. This appendix is included to eliminate any confusion regarding how each CRC is to be calculated.

The following format is used for the CRC4 used in the out-of-band flow control, the CRC24 used in the Burst/Idle Control Word, and the CRC32 used on each lane:

- Data is sent into the CRC24 function MSB first from each byte in the order of byte transmission
- The CRC is transmitted on the line with the same format as the data. The MSB of the MSByte (i.e. the $X^4/X^{24}/X^{32}$ coefficient) is sent out first
- The CRC is generated as follows:
  - The polynomial is reset to all ones
  - The data stream is sent through the polynomial function
  - The polynomial is inverted and transmitted in the bit order defined above

To facilitate clarity, the following data burst and subsequent CRC24 are shown. This first eight words are the data payload, with the ninth word the control word. All values are in hexadecimal format, and bit order goes from left to right, with the leftmost character representing bits [63:60] of the word, and the rightmost character representing bits [3:0]. The CRC24 is the rightmost six characters in the ninth word, shown in bold below - 0x59E69D. The 64B/67B framing bits have been omitted.

Data:

```
520bb1047d585e00
c2b4b401bbaf0100
0000fcb0b3a8468e
1a0a01e1ba38a9df
00003677eea56dda
beb48d4d93a88a12
00001f9515f655dc
c3857a641b260c51
```

Control:

```
f100000000 59e69d
```

Scrambler:

The following Verilog sample code is offered to illustrate the process of implementing and applying the scrambler function. Note that the initial value chosen here is arbitrary, but different values per lane are recommended.

```
module scrambler (
        clk,
        reset,
        lane_number,
        word_is_scrambler_state,
        word_is_synchronization,
        word_is_to_be_scrambled,
        data_in,
        Data
);
```

```
input    clk;
input    reset;
input    [3:0] lane_number;

input    word_is_scrambler_state;
input    word_is_synchronization;
input    word_is_to_be_scrambled;

input    [63:0] data_in;
output   [63:0] Data;

reg      [63:0] Data;
reg      [57:0] Poly;
wire     [63:0] next;

assign next[63] = Poly[57] ^ Poly[38];
assign next[62] = Poly[56] ^ Poly[37];
assign next[61] = Poly[55] ^ Poly[36];
assign next[60] = Poly[54] ^ Poly[35];
assign next[59] = Poly[53] ^ Poly[34];
assign next[58] = Poly[52] ^ Poly[33];
assign next[57] = Poly[51] ^ Poly[32];
assign next[56] = Poly[50] ^ Poly[31];
assign next[55] = Poly[49] ^ Poly[30];
assign next[54] = Poly[48] ^ Poly[29];
assign next[53] = Poly[47] ^ Poly[28];
assign next[52] = Poly[46] ^ Poly[27];
assign next[51] = Poly[45] ^ Poly[26];
assign next[50] = Poly[44] ^ Poly[25];
assign next[49] = Poly[43] ^ Poly[24];
assign next[48] = Poly[42] ^ Poly[23];
assign next[47] = Poly[41] ^ Poly[22];
assign next[46] = Poly[40] ^ Poly[21];
assign next[45] = Poly[39] ^ Poly[20];
assign next[44] = Poly[38] ^ Poly[19];
assign next[43] = Poly[37] ^ Poly[18];
assign next[42] = Poly[36] ^ Poly[17];
assign next[41] = Poly[35] ^ Poly[16];
assign next[40] = Poly[34] ^ Poly[15];
assign next[39] = Poly[33] ^ Poly[14];
assign next[38] = Poly[32] ^ Poly[13];
assign next[37] = Poly[31] ^ Poly[12];
assign next[36] = Poly[30] ^ Poly[11];
assign next[35] = Poly[29] ^ Poly[10];
assign next[34] = Poly[28] ^ Poly[9];
assign next[33] = Poly[27] ^ Poly[8];
assign next[32] = Poly[26] ^ Poly[7];
assign next[31] = Poly[25] ^ Poly[6];
assign next[30] = Poly[24] ^ Poly[5];
assign next[29] = Poly[23] ^ Poly[4];
assign next[28] = Poly[22] ^ Poly[3];
assign next[27] = Poly[21] ^ Poly[2];
assign next[26] = Poly[20] ^ Poly[1];
assign next[25] = Poly[19] ^ Poly[0];
assign next[24] = Poly[57] ^ Poly[38] ^ Poly[18];
assign next[23] = Poly[56] ^ Poly[37] ^ Poly[17];
assign next[22] = Poly[55] ^ Poly[36] ^ Poly[16];
assign next[21] = Poly[54] ^ Poly[35] ^ Poly[15];
assign next[20] = Poly[53] ^ Poly[34] ^ Poly[14];
assign next[19] = Poly[52] ^ Poly[33] ^ Poly[13];
assign next[18] = Poly[51] ^ Poly[32] ^ Poly[12];
```

```
assign next[17] = Poly[50] ^ Poly[31] ^ Poly[11];
assign next[16] = Poly[49] ^ Poly[30] ^ Poly[10];
assign next[15] = Poly[48] ^ Poly[29] ^ Poly[9];
assign next[14] = Poly[47] ^ Poly[28] ^ Poly[8];
assign next[13] = Poly[46] ^ Poly[27] ^ Poly[7];
assign next[12] = Poly[45] ^ Poly[26] ^ Poly[6];
assign next[11] = Poly[44] ^ Poly[25] ^ Poly[5];
assign next[10] = Poly[43] ^ Poly[24] ^ Poly[4];
assign next[9] = Poly[42] ^ Poly[23] ^ Poly[3];
assign next[8] = Poly[41] ^ Poly[22] ^ Poly[2];
assign next[7] = Poly[40] ^ Poly[21] ^ Poly[1];
assign next[6] = Poly[39] ^ Poly[20] ^ Poly[0];
assign next[5] = Poly[57] ^ Poly[19];
assign next[4] = Poly[56] ^ Poly[18];
assign next[3] = Poly[55] ^ Poly[17];
assign next[2] = Poly[54] ^ Poly[16];
assign next[1] = Poly[53] ^ Poly[15];
assign next[0] = Poly[52] ^ Poly[14];

always @(posedge clk) if(reset) begin
        Poly <= {{54{1'b1}}, lane_number[3:0]};//reset each lane differently
        Data <= 64'b0;
end else if(word_is_to_be_scrambled) begin
        Poly <= next[57:0];
        Data <= data_in[63:0] ^ {Poly[57:0], next[63:58]};
end else if(word_is_synchronization) begin
        Data <= 64'h78f678f678f678f6;
end else if(word_is_scrambler_state) begin
        Data <= {6'b001010 , Poly[57:0]};
end

endmodule
```

Appendix C Interoperability Checklist

Interlaken defines a framework for many possible implementations. The following checklist is offered as a guideline for specifying relevant parameters to ensure that two independent implementations may interoperate.

Table 13 Interoperability Checklist

Parameter	Options	Value
Number of Lanes	1 : no inherent limit	
SerDes Rate	No inherent limit	
Number of Channels	1 - 64K	
Transmission Format	Segment-mode, Packet-mode, or both	
Receive Format	Segment-mode, Packet-mode, or both	
Upper Limit of BurstMax	64 : no limit	
BurstShort Requirement	32: no limit	
Scheduling Enhancement (Section 5.3.2.1.1, *Optional Scheduling Enhancement*, on page 14) Supported	Yes or No	
If Scheduling Enhancement (Section 5.3.2.1.1) Supported, Range of BurstMin	32: no limit	
Flow Control Method	In-Band or Out-of-Band	
If Packet-mode, Flow Control Interpretation	When XOFF, stop current packet mid-stream, or finish packet before stopping	
If In-Band, re-use Multiple Use field	Yes or No	
If Out-of-Band, pad technology	LVDS or LVCMOS	
Channel Calendar	Mapping of channels to flow control status slots	
Status Messaging	Yes or No	
Rate Matching Required	Yes or No; if Yes, desired granularity	
Meta Frame Length range	Upper and Lower Bound, in 8- byte Words	

Contact Information

For Cortina:

Address: c/o Interlaken Inquiries,
Cortina Systems,
840 West California Avenue, Suite 100
Sunnyvale, California 94086

Phone: 408 - 481 - 2300

E-Mail: interlaken@cortina-systems.com
Web Site: http://www.cortina-systems.com

For Cisco:

Address: c/o Interlaken Inquiries,
Cisco Systems,
170 Tasman Drive
San Jose, California 95134

Phone: 408 - 526 - 4000

E-Mail: interlaken@external.cisco.com
Web Site: http://www.cisco.com

~ End of Document ~