Working Group: **Physical and Link Layer (PLL)**

TITLE: **System Packet Interface Level 5 (SPI-5): OC-768 System Interface for Physical and Link Layer Devices.**

SOURCE:

Karl Gass
Technical Editor
Sandia National Laboratories
PO Box 5800, MS 0874
Albuquerque, NM 87185 USA
Phone: +1 505 844-8849
Email: kgass@sandia.gov

Russ Tuck
PLL Working Group Chair
Pluris Terabit Network Systems
10455 Bandley Drive
Cupertino, CA 95014 USA
Phone: +1 408 861-3360
Email: tuck@pluris.com

Jeffrey Lynch
PLL Working Group Vice Chair
IBM
P.O. Box 12195
Research Triangle Park,
NC 27709 USA
Phone: +1 919-254-4454
Email: jjlynch@us.ibm.com

Document Status: Implementation Agreement: OIF-SPI5-01.1
Project: SPI-5
DATE: September 2002

Abstract: This contribution contains the Implementation Agreement for an interface between the Physical Layer devices and the Link Layer devices (System Packet Interface or SPI). SPI-5 is an interface for packet and cell transfer between a physical layer (PHY) device and a link layer device, for aggregate bandwidths of OC-768 ATM and Packet over SONET/SDH (POS), as well as other applications at the 40 Gb/s data rate.

For additional information contact:
The Optical Internetworking Forum, 39355 California Street, Suite 307, Fremont, CA 94538
510-608-5928 phone ✦ info@oiforum.com

1 TABLE OF CONTENTS

2 LIST OF FIGURES

3 LIST OF TABLES

4 DOCUMENT REVISION HISTORY

OIF2000.293 Nov 07, 2000
PMC-Sierra proposal for SPI-5 presented at the November 2000 OIF meeting.

OIF2001.134.0 Feb 01,2001
SPI-5 Implementation Agreement Draft

OIF2001.134.1 Apr 23, 2001
SPI-5 Implementation Agreement Draft 2.0

OIF2001.134.8 May 3, 2001
SPI-5 Implementation Agreement Straw Ballot

OIF2001.134.9 July 23, 2001
SPI-5 Implementation Agreement Draft 3.0

OIF2001.134.10 Aug 28, 2001
SPI-5 Implementation Agreement pre-Straw Ballot 2

OIF2001.134.11 Sep 16, 2001
SPI-5 Implementation Agreement Straw Ballot 2

OIF2001.134.12 Oct 29, 2001
SPI-5 Implementation Agreement post-Straw Ballot 2

OIF2001.134.13 Nov 6, 2001
SPI-5 Implementation Agreement (modified Fig 9.4)

OIF-SPI5-01.0 (OIF2001.134.14) Nov 6, 2001
SPI-5 Implementation Agreement (added A,B,C,D reference points to Fig 7.1)

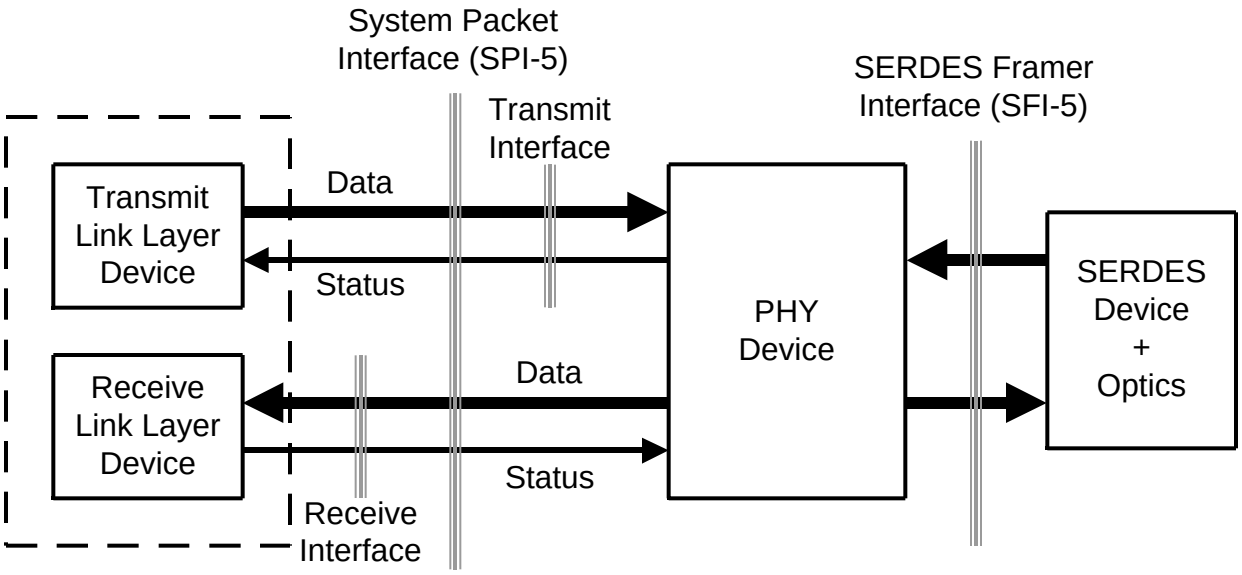
OIF-SPI5-01.1 (OIF2001.134.15) Sep 5, 2002
Maintenance update to fix RSTAT clocking (pg.13)

5 INTRODUCTION

SPI-5 is an interface for packet and cell transfer between a physical layer (PHY) device and a link layer device, for aggregate bandwidths of OC-768 ATM and Packet over SONET/SDH (POS), as well as other applications at the 40 Gb/s data rate. This section provides a general overview of the interface. The following sections contain more detailed descriptions of the signals and associated operations, data structures, and start-up. Appendices have been included to provide guidance on implementations; they are for information only and are not normative parts of this specification.

The following is a general synopsis of the SPI-5 interface. For reference, a general block diagram is shown in Figure 5.1. SPI-5 is the system packet interface for data transfer between the link layer and the PHY device. It is designed to meet requirements of this particular application, although it may be used in other applications as well. "Transmit" and "Receive" refer, respectively, to data flow and associated control/status information for the Link Layer to PHY, and the PHY to Link Layer directions.

Figure 5.1: System Reference Model



Each direction of the SPI-5 bus has its own independent Pool Status Channel. The Transmit Pool Status Channel (TSTAT) reports pool status in the PHY device back to the Link Layer device. The Receive Pool Status Channel (RSTAT) reports pool status in the Link Layer device back to the PHY device. On both the

Transmit and Receive interfaces, Pool status information is sent separately from the corresponding data path. By taking Pool status information out-of-band, it is possible to de-couple the Transmit and Receive interfaces so that each operates independently of the other. Such an arrangement makes SPI-5 suitable not only for bi-directional but also for unidirectional link layer devices. In both the Transmit and Receive interfaces, the Port Address, delineation information and error control coding are sent in-band with the data.

SPI-5 has the following general characteristics:

- Point-to-point connection (i.e., between single PHY and single Link Layer device).
- Support for 256 ports with address extension to 2^{144} ports.
- Transmit / Receive Data Path:
  - Control Words carry In-band Port Address, start/end-of-packet indication and error-control code.
  - Data transfer segmented in bursts that are multiples of 16 words (32 bytes) in length.
  - 2.488 Gb/s minimum data rate per line on data path
  - Independent $X^{11} + X^9 + 1$ stream cipher scrambling on each bit lane.
  - Source synchronous clock at one quarter the data bit rate that acts solely as data recovery frequency reference.
- Transmit / Receive Pool Status Channel:
  - Operates at the same clock rate as the data path
  - Bit serial Pool status indication
  - In-band pool status framing
  - Electrical, training and scrambling functions common with data path
  - In-band training
- Electrical characteristics are included in the Common Electrical Specification (SxI-5)

6 INTERFACE DESCRIPTION

This section provides a brief introduction of the SPI-5 bus. Section 7 contains signal definitions for the Transmit and Receive directions. Section 8 describes the signal operation along with the data structures for payload data and in-band control/status information. Flow control mechanisms are described in Section 9. Bus configuration parameters are described in Section 10, while system start-up procedures are detailed in Section 11.

SPI-5 transfers four words on the data path and four status report bits on the Pool Status Channel for each word cycle (Wcycle). A word on the SPI-5 bus may be either a Control word or a Data word. A high logic level on the associated xCTL signal denotes a Control word. It may contain the status of a previous burst of Data words, the Port Address of a subsequent burst of Data words, and an error checksum on the previous burst of Data words plus the Control word itself. It may also be an Idle control word used to fill in dead-time on the bus when no data or control information needs to be transferred. A low logic level on the associated TCTL or RCTL signal denotes a Data word. A Data word may contain packet payload data or port address bytes. A special Training sequence of Control and Data words is used to allow the sink device to adapt to skews over the various signals that constitute the bus.

Payload data is sent over the SPI-5 bus in Transfers containing multiples of 16 data words (32 bytes). A Transfer will also terminate upon an end-of-packet Control Word. At least one Control word must immediately precede a Transfer and carry the start-of-packet status and the address of the port associated with the Transfer. A Control word must immediately follow a Transfer and contain the end-of-packet status. Two contiguous Transfers may share a common Control word whose fields refer both to the previous Transfer and to the subsequent Transfer.

7 SIGNALS

A block diagram depicting the interface signals is shown in Figure 7.1. The Transmit and Receive data paths include, respectively, (TDCLK, TDAT[15:0], TCTL) and (RDCLK, RDAT[15:0], RCTL). The Transmit and Receive Pool status channels include TSTAT, and RSTAT, respectively. A,B,C, and D define reference points for the Common Electrical Specification (SxI-5).

Figure 7.1: SPI-5 Interface

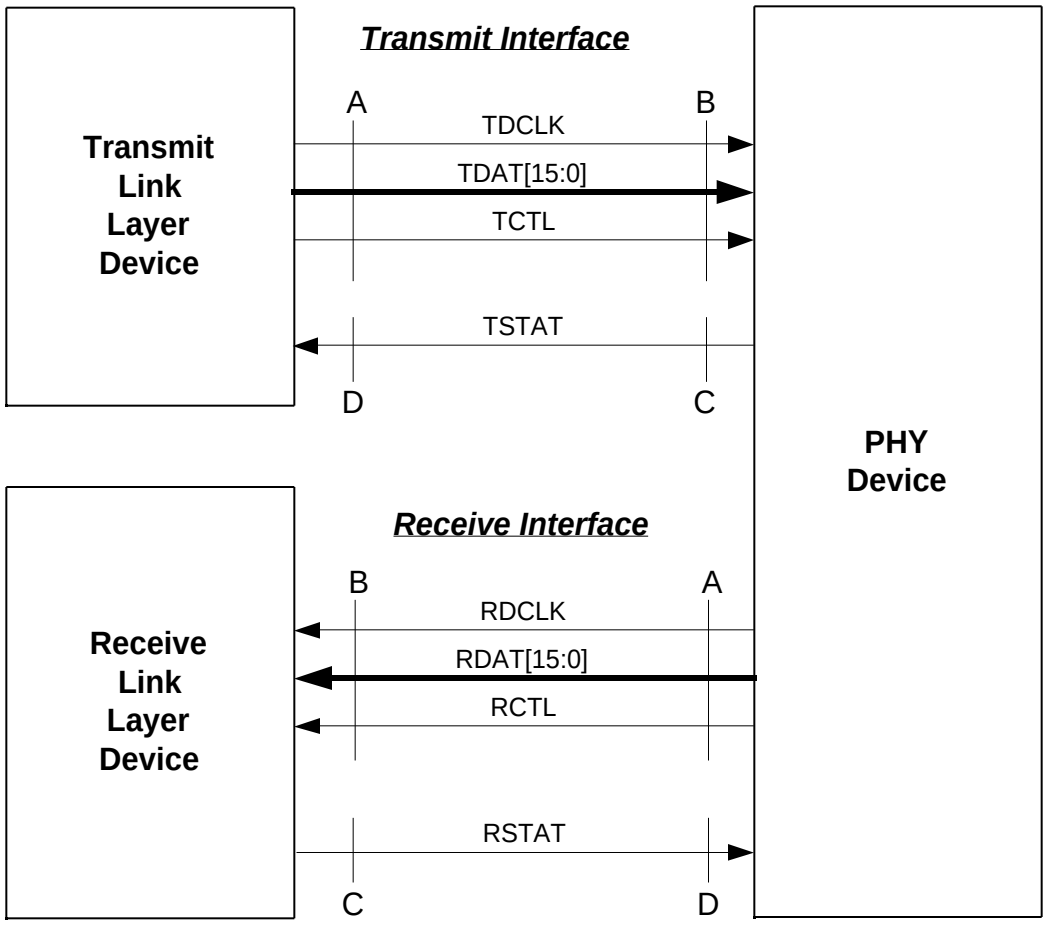


Table 7.1: SPI-5 Interface Signal Summary

Signal	Direction	Description
TDCLK	Link to PHY	Transmit Data Clock The Transmit Data Clock signal (TDCLK) provides timing reference for the Transmit data path signals (TDAT, TCTL). TDCLK is nominally a 50% duty cycle clock with a frequency that is one-quarter of the data bit rate of TDAT and TCTL. TDCLK is frequency locked to these signals. The minimum clock frequency is 622 MHz. Static phase offset between TDCLK and TDAT, and TCTL is unspecified. The source device of the Transmit interface shall generate TDCLK. It is optional for the sink device in the Transmit interface to use TDCLK. If the sink device does not use TDCLK, the source device is permitted to disable TDCLK.
TDAT[15:0]	Link to PHY	Transmit Data The Transmit Data bus (TDAT) carries Data and Control words from the Link Layer to the PHY device. TDAT[15:8] carries the last byte of an odd byte length packet, while TDAT[7:0] carries the last byte of an even byte length packet.The minimum data rate of each bit lane is 2.488 Gb/s. TDAT is frequency locked to TDCLK with unspecified static phase offset.
TCTL	Link to PHY	Transmit Control The Transmit Control signal (TCTL) identifies control words on the Transmit Data bus (TDAT). TCTL is set high when a Control word is present on TDAT[15:0]. It is set low for Data words. The minimum data rate of TCTL is 2.488 Gb/s. TCTL is frequency locked to TDCLK with unspecified static phase offset.

Signal	Direction	Description
TSTAT	PHY to Link	Transmit Status Channel The Transmit Status Channel signal (TSTAT) carries Pool status information, along with associated error detection and framing. TSTAT reports the status of all Calendar entries. The minimum data rate of TSTAT is 2.488 Gb/s. TSTAT is frequency locked to TDCLK with unspecified static phase offset.
RDCLK	PHY to Link	Receive Data Clock The Receive Data Clock signal (RDCLK) provides timing reference for the receive data path signals (RDAT and RCTL). RDCLK is nominally a 50% duty cycle clock with a frequency that is one-quarter of the data bit rate of RDAT and RCTL. RDCLK is frequency locked to these signals. The minimum clock frequency is 622 MHz. Static phase offset between RDCLK and RDAT, and RCTL is unspecified. The source device of the Receive interface shall generate RDCLK. It is optional for the sink device in the Receive interface to use RDCLK. If the sink device does not use RDCLK, the source device is permitted to disable RDCLK.
RDAT[15:0]	PHY to Link	Receive Data The Receive Data bus (RDAT) carries payload data and in-band control from the PHY to the Link Layer device. RDAT[15:8] carries the last byte of an odd byte length packet, while RDAT[7:0] carries the last byte of an even byte length packet.The minimum data rate of each bit lane is 2.488 Gb/s. RDAT is frequency locked to RDCLK with unspecified static phase offset.

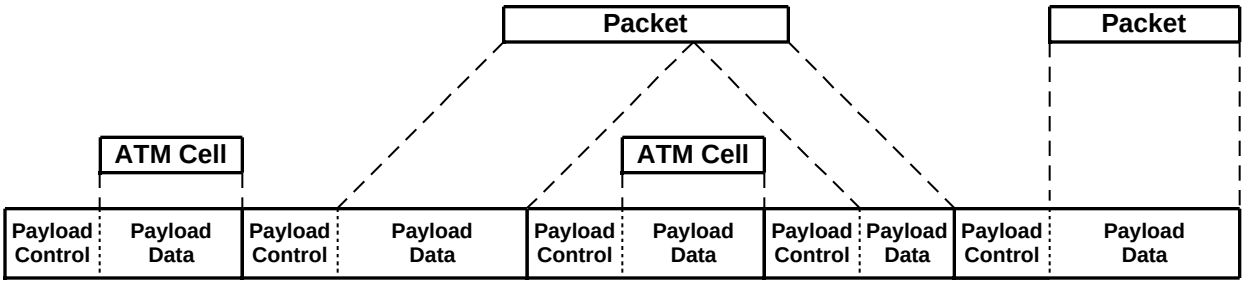
Signal	Direction	Description
RCTL	PHY to Link	Receive Control The Receive Control signal (RCTL) identifies control words on the Receive Data bus (RDAT). RCTL is set high when a Control word is present on RDAT[15:0]. It is set low for Data words. The minimum data rate of RCTL is 2.488 Gb/s. RCTL is frequency locked to RDCLK with unspecified static phase offset.
RSTAT	Link to PHY	Receive Pool Status The Receive Status bus (RSTAT) carries Pool status information, along with associated error detection and framing. RSTAT reports the status of all Calendar entries. The minimum data rate of RSTAT is 2.488 Gb/s. RSTAT is frequency locked to RDCLK with unspecified static phase offset.

8 DATA PATH OPERATION

8.1 Data Transfer

Data is delivered over the SPI-5 bus in bursts called Transfers. The start of a packet is aligned to the beginning of a Transfer. Data from disparate packets will not share a Transfer. Transfers terminate after a multiple of 32 bytes has been sent or at an end-of-packet. Packets may be segmented into two or more Transfers. All Transfers for that packet, except the last, must have multiples of 32 bytes of payload. The last Transfer may be shorter if the total packet length is not a multiple of 32 bytes. The Transfers making up a packet do not all need to be the same length. Figure 8.1 shows the segmentation of packets and cells into Transfers on the payload stream.

Figure 8.1: Segmentation of Packets and Cells

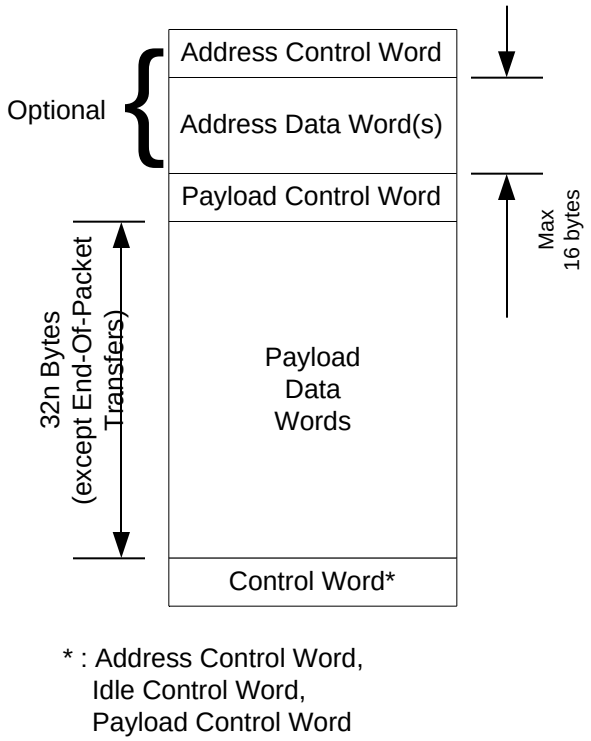


Transfers consist of a burst of data words and are delimited by Control words. (See Figure 8.2.)

Once a Transfer has begun, data words are sent in an uninterrupted burst until the Transfer terminates at an end-of-packet or after a multiple of 32 bytes has been sent. An optional set of Address Control and Address Data words and a mandatory Payload Control word precede each Transfer. The Address Control and Address Data words carry the least significant bytes of the Port Address. The Payload Control word ahead of each Transfer carries the start-of-packet flag and the most significant byte of the Port Address. The Control word following each transfer carries the end-of-packet status and a DIP-4 code as error checksum, except that the DIP-4 may be omitted from the following Address Control word in certain cases (see Table 8.7). In the latter case the subsequent Payload Control word carries the DIP-4 for a transfer immediately preceding the Address Control word. For back-to-back transfers, the Control word ending the previous Transfer and the one starting the next Transfer may be shared. Certain fields within the Control word would refer to the previous and others would refer

to the next transfer. During idle periods, where no data delivery is required, Idle Control words are sent. Control words are described in detail in Section 8.4.

Figure 8.2: Anatomy of a Transfer



* : Address Control Word,
Idle Control Word,
Payload Control Word

8.1.1 Burst Admission Procedure

The length of a burst data Transfer is nominally a multiple of 32-byte blocks. However, at the end of a packet a burst data Transfer can be any length needed, including a short burst of fewer than 32 bytes. This allows an SPI-5 interface to efficiently transfer packets of any length.

Many (if not all) SPI-5 devices operate internally at lower clock rates than the external SPI-5 signals. In such devices full bandwidth operation requires the internal bus carrying signals to or from an SPI-5 interface to be wider than the external bus. This implies that burst boundaries can occur at more than one position on the internal bus. To facilitate further processing, these bursts may be separated and realigned. This introduces some inefficiency in the use of the internal bus, because it effectively rounds bursts up to the nearest multiple of the internal bus size. Consequently it is very difficult to design a data path that can handle a long stream of short burst Transfers.

Although the output side of an SPI-5 data interface has the freedom to insert idle control words as needed to accommodate internal bus alignment constraints, the input side of the interface must accommodate any legal burst transfer generated by the sender. The Burst Admission Procedure determines when Idle Control Words must be inserted by the sender so that the input data path is not overwhelmed by continuous short burst transfers.

The Burst Admission Procedure regulates consecutive burst transfers across a data interface, regardless of Credit Pools, to avoid overloading the input data path bandwidth. By comparison the Pool Status (described in Section 9) regulates burst transfers on a per-Pool basis, regardless of sending order, to avoid overloading the input data FIFO for the Pool. In order for a burst transfer to begin, both the Burst Admission Procedure and the Pool Status must allow it.

The Burst Admission Procedure uses what is commonly known as a Token Bucket algorithm at the output side of the interface. This algorithm uses per-block tokens that are generated at a fixed rate, stored in a finite container and consumed by burst transfers. Each token represents the ability to transfer a single block of 16 or fewer Payload Data Words or a block of 8 or fewer Address Data Words. Properly matching the rate of token generation and the maximum number of accumulated tokens to the design of the input data path ensures that its capacity can be utilized but not exceeded.

Tokens are stored in an up/down counter that can hold from 0 to TOKEN_MAX tokens, where TOKEN_MAX is a bus parameter (see Table 10.1). When a token is generated the counter is incremented; when a token is consumed the counter is decremented. Increments cannot cause the counter to exceed TOKEN_MAX. Decrements cannot cause the counter to fall below zero. The value of TOKEN_MAX is a design parameter at the input side of the interface and a provisionable parameter at the output side of the interface. The minimum value of TOKEN_MAX is $\mathbf{min((2N_p - 1), 7)}$, where N_p is the number of Credit Pools**.**

One token is generated for every TOKEN_GEN word cycles, where TOKEN_GEN is a bus parameter (see Table 10.1). The value of TOKEN_GEN is chosen so that tokens are generated faster than once per 32-byte block of SPI-5 data path speed. This ensures that once a multi-block burst transfer starts it cannot be interrupted. The value of TOKEN_GEN is a design parameter at the input side of the interface and a provisionable parameter at the output side of the interface. The maximum value of TOKEN_GEN is 12 word cycles (24 bytes).

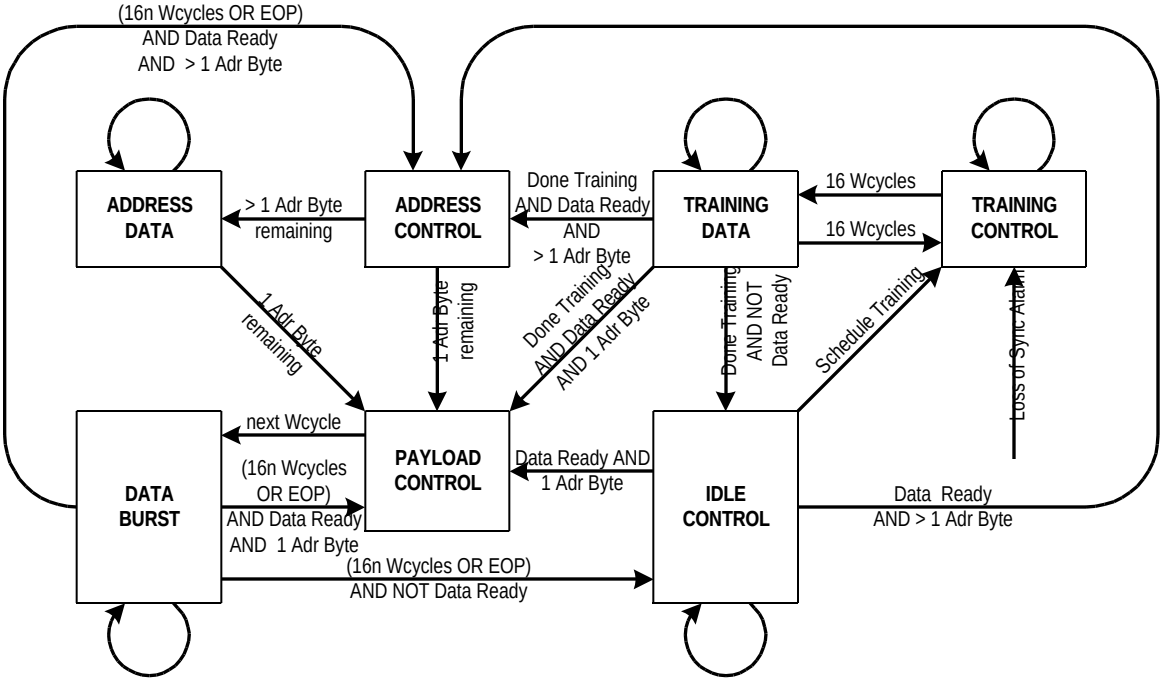
Before sending a burst of Address Data Words and Payload Data Words the sender must reserve one or two tokens. One token is consumed for every block of 16 or fewer Payload Data Words sent. One additional token is consumed for every block of 8 or fewer Address Data Words sent. Thus, for example, a burst of 3 Payload Data Words requires one token, a burst of two Address Data Words

plus 12 Payload Data Words requires two tokens, and a burst of 300 Payload Data Words requires one token to start and consumes 18 additional tokens to complete.

8.1.2 Data Transfer Procedure

The proper sequence of Control and Data words on the data path constitues a state machine. A state transition occurs for each word cycle (Wcycle). Figure 8.3 shows a Wcycle-by-Wcycle behavior of the data path state transitions.

Figure 8.3: Data Path State Diagram



For delivery of data over the SPI-5 bus, only states Idle Control, Address Control, Address Data, Payload Control, and Data Burst are involved. The remaining states, Training Control and Training Data, pertain to de-skew of the signals in the bus and will be described further in Section 8.8. Table 8.1 below provides a description of the states.

Table 8.1: Data Path States

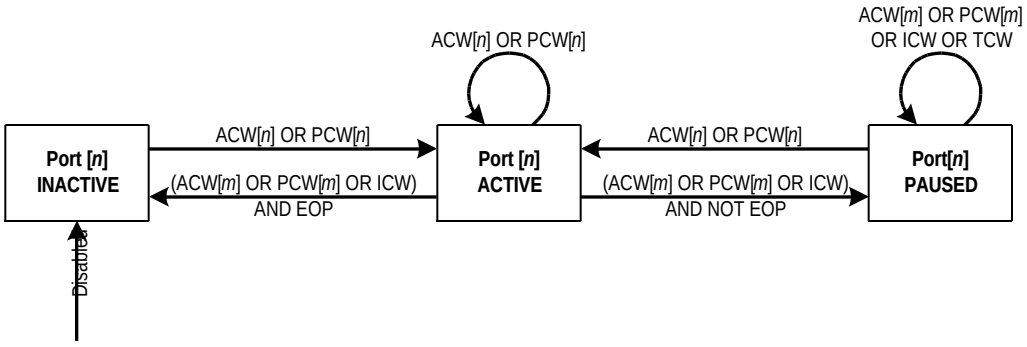
State	Description
Idle Control	This state handles the case where no data needs to be sent over the bus or to insert padding between bursts as

State	Description
	required by the burst admission procedure. An Idle Control word is sent in the current Wcycle.
Address Control	This state is entered at the beginning of each Transfer that sends more than one byte of Port Address. An Address Control word is sent to deliver the least significant byte of the Port Address. The more significant bytes of the Port Address are located in subsequent Address Data words and a Payload Control word.
Address Data	This state is entered after the Address Control state to deliver the intermediate words of the Port Address in a sequence of Address Data words. Each Address Data word contains 2 bytes of Port Address. The words in the sequence carry Port Address bytes from less significant to more significant. Residence in this state is maintained until all but the most significant byte of the Port Address has been sent. The maximum duration in this state is 8 Wcycles. An Address Data word is sent for every Wcycle in this state.
Payload Control	This state is entered at the beginning of each Transfer that has only one Port Address byte. It is also entered after the Address Control or Address Data words to deliver the most significant byte of the Port Address in Transfers that have multiple Port Address bytes. A Payload Control word is sent in this state.
Data Burst	The payload data of a Transfer is sent in this state. Residence is maintained until the scheduled number of bytes of data, for the current Transfer, has been sent or an end-of-packet is encountered. A Data word is sent for every Wcycle in this state.
Training Control	This state provides the Control word portion of the deskew training pattern. A Training Control word is sent for every Wcycle in this state. Training is described in detail in section 9.
Training Data	This state provides the Data word portion of the deskew training pattern. A Training Data word is sent for every Wcycle in this state. Training is described in detail in section 9.

The SPI-5 bus has the potential to address an extremely large number of logical ports. A device does not have to implement them all. It must, however, be able to distinguish between all the ports within the Physical Address field without aliasing. The treatment of the remaining Port Address bytes is application specific.

Each port implemented in the device is independent of the others and maintains its own state machine. State transitions occur when Control words are sent on the bus. Transitions can occur in two separate ports (Port[*m*], Port[*n*]) at each Control word. Figure 8.4 shows the state transition diagram for an arbitrary (Port [*n*]) state machine at the source of a Transmit or Receive interface. A port is in the Active state when it is currently originating a Transfer. When a port needs to send one or more Transfers in order to complete the current packet, it is in the Paused state. A port is in the Inactive state when it has no segments of a partially sent packet outstanding. In Figure 8.4, a Control word addressed to Port [*n*] is denoted by the index *n* associated with the word. Control words addressed to other ports are represented by the index *m*.

Figure 8.4: Port State Diagram - Interface Source

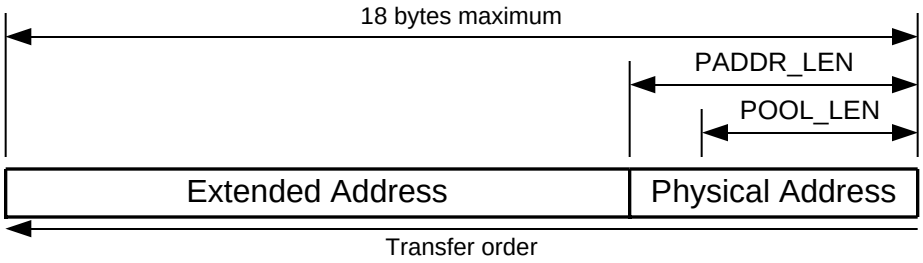


8.2 Address Mapping

Figure 8.5 shows the organization of Port Addresses for the SPI-5 bus. A rich address space is supported. Applications are free to define fields within the Port Address for network segregation, priority indications, etc. The maximum size of a Port Address is 18 bytes, consisting of an Extended Address field and a Physical Address field. The Physical Address field is located in the lower order bytes and has a length that is configured by the bus parameter PADDR_LEN. The maximum range of physical addresses a device is capable of supporting is governed by the TOP_PADDR bus parameter. The Extended Address field is located in the more significant bytes. All Transfers that are part of the same packet must have the same Port Address.

The source and sink devices do not need a common bus parameter to define the length of the Port Address. Conceptually, at the start of each Transfer, the extended address field of the Port Address is initialized to zero. The sequence of Address Control words, Address Data words and Payload Control words at the start of a Transfer fills in the Port Address from the least significant to most significant byte. The portion of the Extended Address field that has not been sent explicitly will remain at zero. The width, in bytes, of the physical address field is given by PADDR_LEN. Those address bytes are uniquely decoded to locate ports within the sink device. At the start of every Transfer, the physical address field of the port address must be sent explicitly. The extended address field may be partially specified. The remaining bytes are implicitly set to zero. If the source sends fewer address bytes than the sink can handle, the unsent higher order bytes are treated as all-zeros. On the other hand, if the source sends a larger number of address bytes than the sink can handle, the sink will ignore the more significant bytes beyond its own limit.

Figure 8.5: Port Address Format



8.3 Payload Mapping

Examples of transferring ATM cells and packets are shown below.

Table 8.2 shows the mapping of ATM cell to a Transfer using a 52-byte structure. The HEC field in the ATM cell header is omitted. The first header byte (H1) is placed in the more significant byte of the first Data word (DW[15:8]) while the second header byte (H2) is placed in the less significant byte (DW[7:0]). Similarly, the first cell payload byte (P1) is placed in the more significant byte of the Data word (DW[15:8]) while the second payload byte (P2) is placed in the less significant byte (DW[7:0]). The start of a cell is always aligned to the start of a Transfer. A Transfer terminates at the end of a cell. Consequently, two cells will not share the Transfer. For illustrative purposes, four Port Address bytes are shown: one in the Address Control word (ACW), two in the Address Data word (ADW) and the fourth in the Payload Control word (PCW). The address bytes carried in the Address Control words form the least significant byte of the Port Address while the address byte in the Payload Control word forms the most significant byte.

Table 8.2: 52-byte ATM Cell Mapping Structure

Wcycle	Word Type	*x*DAT[15:8]	*x*DAT[7:0]	Description
1	Address Control	ACW[15:8]	ACW[7:0]	Port Address [7:0]
2	Address Data	PA[23:16]	PA[15:8]	Port Address [23:8]
3	Payload Control	PCW[15:8]	PCW[7:0]	Port Address [31:24]; Start-Of-Packet
4	Payload Data	H1[7:0]	H2[7:0]	ATM header H1, H2
5	Payload Data	H3[7:0]	H4[7:0]	ATM header H3, H4
6	Payload Data	P1[7:0]	P2[7:0]	ATM payload P1, P2
7 to 28	Payload Data	—	—	ATM payload P3 to P46
29	Payload Data	P47[7:0]	P48[7:0]	ATM payload P47, P48
30	Any Control	*x*CW[15:8]	*x*CW[7:0]	End-Of-Packet; 2 bytes valid

Table 8.3 shows the mapping of ATM cell to a Transfer using a 54-byte structure. The HEC field in the ATM cell header is padded with a User Defined Field (UDF). The first header byte (H1) is placed in the more significant byte of the first Data word (DW[15:8]) while the second header byte (H2) is placed in the less significant byte (DW[7:0]). Similarly, the first cell payload byte (P1) is placed in the more significant byte of the Data word (DW[15:8]) while the second payload byte (P2) is placed in the less significant byte (DW[7:0]). The start of a cell is always aligned to the start of a Transfer. A Transfer terminates at the end of a cell. Consequently, two cells will not share the Transfer. For illustrative purposes, four Port Address bytes are shown: one in the Address Control word (ACW), two in the Address Data word (ADW), and the fourth in the Payload Control word (PCW). The address bytes carried in the Address Control words form the least significant bytes of the Port Address while the address byte in the Payload Control word forms the most significant byte.

Table 8.3: 54-byte Cell Mapping Structure

Wcycle	Word Type	*x*DAT[15:8]	*x*DAT[7:0]	Description
1	Address Control	ACW[15:8]	ACW[7:0]	Port Address [7:0]
2	Address Data	PA[23:16]	PA[15:8]	Port Address [23:8]

Wcycle	Word Type	*x*DAT[15:8]	*x*DAT[7:0]	Description
3	Payload Control	PCW[15:8]	PCW[7:0]	Port Address [31:24]; Start-Of-Packet
4	Payload Data	H1[7:0]	H2[7:0]	ATM header H1, H2
5	Payload Data	H3[7:0]	H4[7:0]	ATM header H3, H4
6	Payload Data	HEC[7:0]	UDF[7:0]	ATM HEC and UDF
7	Payload Data	P1[7:0]	P2[7:0]	ATM payload P1, P2
8 to 29	Payload Data	—	—	ATM payload P3 to P46
30	Payload Data	P47[7:0]	P48[7:0]	ATM payload P47, P48
31	Any Control	*x*CW[15:8]	*x*CW[7:0]	End-Of-Packet; 2 bytes valid

Table 8.4 shows the mapping of packet with an odd number of bytes into a single Transfer. The start of a packet is always aligned to the start of a Transfer. The end of a packet terminates a Transfer. Thus, two packets will not share a Transfer. The first payload byte (P1) is placed in the more significant byte of the Data word (DW[15:8]) while the second payload byte (P2) is placed in the less significant byte (DW[7:0]). Packets may contain either an odd or and even number of bytes. In the case of an odd byte count, the last Data word would only carry one valid byte in the most significant bits. The least significant byte must be set to 00 hex. The Control word terminating the Transfer will indicate end-of-packet with a single valid byte in the previous Data word. For illustrative purposes, six Port Address bytes are shown one in the Address Control word (ACW), four in the Address Data words (ADW) and the sixth in the Payload Control word (PCW). The address bytes carried in the Address Control word form the least significant byte of the Port Address while the address byte in the Payload Control word form the most significant byte. The length of the packet in Table 8.4 is chosen to be 49 bytes for illustrative purposes only. All Transfers of packets with an odd number of bytes will have the same mapping.

The least significant byte must be set to 00 hex.

Table 8.4: Packet Mapping Structure - Odd Byte Count

Wcycle	Word Type	*x*DAT[15:8]	*x*DAT[7:0]	Description
1	Address Control	ACW[15:8]	ACW[7:0]	Port Address [7:0]

Wcycle	Word Type	*x*DAT[15:8]	*x*DAT[7:0]	Description
2	Address Data	PA[23:16]	PA[15:8]	Port Address [23:8]
3	Address Data	PA[39:32]	PA[31:24]	Port Address [39:24]
4	Payload Control	PCW[15:8]	PCW[7:0]	Port Address [47:40]; Start-Of-Packet
5	Payload Data	P1[7:0]	P2[7:0]	Packet payload P1, P2
6 to 28	Payload Data	—	—	Packet payload P3 to P48
29	Payload Data	P49[7:0]	00 hex	packet payload P49, pad
30	Any Control	xCW[15:8]	xCW[7:0]	End-Of-Packet; 1 byte valid

Table 8.5 shows the mapping of packet with an even number of bytes into a single Transfer. The start of a packet is always aligned to the start of a Transfer. The end of a packet terminates a Transfer. Thus, two packets will not share a Transfer. The first payload byte (P1) is placed in the more significant byte of the Data word (DW[15:8]) while the second payload byte (P2) is placed in the less significant byte (DW[7:0]). Packets may contain either an odd or and even number of bytes. In the case of an even byte count, the last Data word would carry two valid bytes. The Control word terminating the Transfer will indicate end-of-packet with two valid bytes in the previous Data word. For illustrative purposes, two Port Address bytes are available, one in the Address Control words (ACW) and the other in the Payload Control word (PCW). The address byte carried in the Address Control word form the less significant byte of the Port Address while the address byte in the Payload Control word form the more significant byte. The length of the packet in Table 8.5 is chosen to be 50 bytes for illustrative purposes only. All Transfers of packets with an even number of bytes will have the same mapping.

Table 8.5: Packet Mapping Structure - Even Byte Count

Wcycle	Word Type	*x*DAT[15:8]	*x*DAT[7:0]	Description
1	Address Control	ACW[15:8]	ACW[7:0]	Port Address [7:0]
2	Payload Control	PCW[15:8]	PCW[7:0]	Port Address [15:8]; Start-Of-Packet

Wcycle	Word Type	*x*DAT[15:8]	*x*DAT[7:0]	Description
3	Payload Data	P1[7:0]	P2[7:0]	Packet payload P1, P2
4 to 26	Payload Data	—	—	Packet payload P3 to P48
27	Payload Data	P49[7:0]	P50[7:0]	packet payload P49, P50
28	Any Control	xCW[15:8]	xCW[7:0]	End-Of-Packet; 2 bytes valid

Table 8.6 shows the mapping of a large packet into multiple Transfers. The start of a packet is aligned to the start of a Transfer. The amount of data sent in the first Transfer and in the intermediate Transfers is application specific. The payload byte count, however, must be a multiple of 32 bytes. Once the scheduled amount of data is sent, the Transfer terminates. Data within each Transfer must originate from the same packet. An end-of-packet terminates the final Transfer. Data from two packets will not share a Transfer. For any one port, only one packet may be outstanding at any time. I.e., it is illegal to start a second packet on a port when the first has not yet completed. The first payload byte (P1) is placed in the more significant byte of the Data word (DW[15:8]) while the second payload byte (P2) is placed in the less significant byte (DW[7:0]). For all the Transfers that make up a packet, the Port Address constructed from Address Control, Address Data, and Payload Control words must be identical. For illustrative purposes, the only Port Address byte available is carried in the Payload Control word (PCW). Neither Address Control nor Address Data words are shown.

The example below shows the packet divided into 3 segments. The first two have lengths of *32i* and *32j* bytes, respectively. The last Transfer is terminated by an end-of-packet and is, thus, not limited to carrying multiples of 32 bytes of payload. The example in Table 8.6 shows an even number of bytes in the packet. An odd number is also possible. The only differences exist at the last Data word where the less significant byte is set to 00 hex and at the last Control word reporting end-of-packet would indicate one valid byte instead of two.

The rules for dividing a packet into multiple Transfers also apply to segmenting ATM cells. On a SPI-5 bus, an ATM cell is treated no differently than a packet of 52 bytes (HEC filtered) or of 54 bytes (HEC and UDF). All but the last Transfer must carry multiples of 32 bytes of ATM header/payload bytes. The final Transfer, as in that of packets, would not have a length limit. The Port Address of all the Transfers that are part of a cell must be identical.

Table 8.6: Packet Mapping Structure - Multiple Segments

Wcycle	Word Type	*x*DAT[15:8]	*x*DAT[7:0]	Description
1	Payload Control	PCW[15:8]	PCW[7:0]	Port Address [7:0]; Start-Of-Packet; First segment
2	Payload Data	P1[7:0]	P2[7:0]	Packet payload P1, P2
3 to *16i*	Payload Data	—	—	Packet payload
16i + 1	Payload Data	P*32i-1*[7:0]	P*32i*[7:0]	Last bytes of first segment
16i + 2	Any Control	xCW[15:8]	xCW[7:0]	End-Of-Transfer
16i + 3 to *X*				Transfers to other ports
X + 1	Payload Control	PCW[15:8]	PCW[7:0]	Port Address [7:0]; Start-Of-Transfer Second segment
X + 2	Payload Data	P*32i+1*[7:0]	P*32i+2*[7:0]	Packet payload bytes P*32i+1*, P*32i+1*
X + 3 to *X* + *16j*	Payload Data	—	—	Packet payload
X + *16j* + 1	Payload Data	P*32(i+j)-1* [7:0]	P*32(i+j)* [7:0]	Last bytes of second segment
X + *16j* + 2	Any Control	xCW[15:8]	xCW[7:0]	End-Of-Transfer
X + *16j* + 3 to *Y*				Transfers to other ports
Y + 1	Payload Control	PCW[15:8]	PCW[7:0]	Port Address [7:0]; Start-Of-Transfer Final segment
Y + 2	Payload Data	P*32(i+j)+1* [7:0]	P*32(i+j)+2* [7:0]	Packet payload bytes P*32(i+j)+1*, P*32(i+j)+2*
Y + 3 to *Y* + 11	Payload Data	—	—	Packet payload

Wcycle	Word Type	*x*DAT[15:8]	*x*DAT[7:0]	Description
Y + 12	Payload Data	P*32(i+j)+21* [7:0]	P*32(i+j)+22* [7:0]	Last bytes of packet
Y + 13	Any Control	xCW[15:8]	xCW[7:0]	End-Of-Packet; 2 bytes valid

8.4 Control Words

Control words (CW[15:0]) are denoted by a logic high level on the associated xCTL signal. A common format is used in both the Transmit and Receive interfaces. Table 8.7 describes the fields in the Control word.

Four Control words have been defined; Address Control word (ACW), Idle Control word (ICW), Payload Control word (PCW), and Training Control word (TCW). The Address Control word, in conjunction with Address Data words, allows systems to have Port Address ranges that are greater than Payload Control words alone could provide. The Idle Control word is used to occupy word cycles where neither data nor control information needs to be sent. A Payload Control word always precedes the burst of payload data in every Transfer. Training Control words are used together with Training Data words to deskew the signals on the bus. Table 8.7 shows the fields of a Control word.

Table 8.7: Fields of Control Words

CW Bits	Mnemonic	Description
CW[15,12]	CWT	Control Word Type The Control Word Type field (CWT[1:0]) identifies the type of the Control word. CWT[1:0] = 0 0 : Idle Control Word (ICW) or Training Control Word (TCW) CWT[1:0] = 0 1 : Address Control Word (ACW) CWT[1:0] = 1 0 : Payload Control Word (PCW), not start-of-packet Transfer CWT[1:0] = 1 1 : Payload Control Word (PCW), start-of-packet Transfer CWT[1] is mapped to CW[15] while CWT[0] is mapped to CW[12].

CW Bits	Mnemonic	Description
CW[14:13]	EOPS	End-of-Packet Status The End-of-Packet (EOP) Status field (EOPS[1:0]) reports the packet status of the immediately preceding payload transfer. EOPS[1:0] = 0 0 : Not an EOP. EOPS[1:0] = 0 1 : EOP, Abnormal cell/packet termination, Abort (application-specific error condition). EOPS[1:0] = 1 0 : EOP, Normal cell/packet termination, 2 bytes valid. EOPS[1:0] = 1 1 : EOP, Normal cell/packet termination, 1 byte valid. EOPS[1:0] is only valid in the first Control word (ACW, ICW or PCW) following a burst of Payload Data words. It is unused and set to “0 0” otherwise. EOPS[1:0] is mapped to CW[14:13].
CW[11:4]	ADR	Address The Address field (ADR[7:0]) provides 8 bits of the Port Address of the next Transfer. The ADR[7:0] located in an Address Control word provides the least significant byte of the Port Address. ADR[15:0] in subsequent Address Data words provide additional bytes of the Port Address, from less to more significant. The ADR[7:0] located in a Payload Control word provides the most significant byte of the Port Address. ADR[7:0] must be set to all zeros in Idle Control words and set to all ones in Training Control words. ADR[7:0] is mapped to CW[11:4].

CW Bits	Mnemonic	Description
CW[3:0]	DIP-4	Diagonal Interleaved Parity The 4-bit Diagonal Interleave Parity field (DIP-4[3:0]) provides transmission error checking over the current Control word and any immediately preceding Data words. Odd parity is calculated diagonally over the covered words. The computation of DIP-4 code is detailed in Section 8.6. DIP-4[3:0] is mapped to CW[3:0]. In applications that require Address Control words for all Transfers, the contents of this field is unspecified. In this case the computation of DIP-4 code extends to the subsequent Payload Control word.

Address Control words (ACW) provide extended Port Address bytes to support systems that have a large address range. An ACW and a sequence of Address Data words (ADW) can be concatenated to build up a multi-byte Port Address. The ACW in the sequence provide the least significant byte of the Port Address. Subsequent ADWs provide additional bytes of the Port Address in increasing order of significance. The most significant byte resides in the Payload Control word, which follows immediately. The maximum length of ADWs in a sequence is 8 words (16 bytes). Thus, the maximum Port Address is 18 bytes in length.

An Address Control word may be used to terminate a Transfer. This occurs when a Payload Data word (PDW) was sent in the previous word cycle. The EOPS field describes the end-of-packet status of that Transfer. The DIP-4 field covers the preceding sequence of Data words and the current ACW. When an ACW follows another Control word, the EOPS field is unused and is set to indicate no EOP (EOPS[1:0] = “ 0 0”). The only legal words to follow an ACW are an Address Data word or a Payload Control word. Idle Control words, Training Control words, Payload Data words, and Training Data words are illegal.

A Payload Control word (PCW) indicates that the next word cycle is the beginning of a data burst in a Transfer. The next word must be a Payload Data word. The Address field in the PCW provides the most significant byte of the Port Address of the Transfer.

A Payload Control word may only be used to terminate a Transfer if only one byte of Port Address is needed. The PCW is shared between both Transfers. It terminates the previous Transfer and initiates the next. The final Payload Data word of the previous burst was sent in the prior word cycle and the first PDW of the subsequent burst is sent in the next word cycle. The EOPS field describes the end-of-packet status of the previous Transfer. The DIP-4 field covers the

preceding sequence of Data words and the current PCW. When a PCW follows another Control word, the EOPS field is unused and is set to indicate no EOP (EOPS[1:0] = “0 0”). The only legal word to follow a PCW is a Payload Data word. All Control words, Address Data words, and Training Data words are illegal.

Idle Control words (ICW) indicate that the bus is idle. When there is no data transfer required, the bus will be filled with ICWs. The ADR field of an ICW is unused and is set to all zeros.

An ICW may be used to terminate a Transfer. This occurs when a Data word was sent in the previous word cycle. The EOPS field describes the end-of-packet status of the Transfer. When used, the DIP-4 field covers the preceding sequence of Payload Data words and the current ICW. When an ICW follows another Control word, the EOPS field is unused and is set to indicate no EOP (EOPS[1:0] = “0 0”). The only legal words to follow an ICW are an Address Control word, a Payload Control word, a Training Control Word, or another Idle Control word. All Data words are illegal.

Training Control words (TCW), in conjunction with Training Data words (TDW), are used to allow the sink device to compensate for the skew between signal traces on the bus. The EOPS field is unused and is set to “0 0”. The ADR and DIP-4 fields are similarly unused and are set to all ones. Since training can only be initiated when the bus is in idle (Figure 8.3), the first TCW in the training sequence is preceded by one or more Idle Control words. Thus, a TCW will not be used to terminate a Transfer. The only legal words to follow a TCW are a Training Data word or another Training Control word. For the case where the DIP-4 field in an ACW is unused, the DIP-4 field of the PCW covers the preceding sequence of PDWs, ACW, ADWs and the current PCW. Payload Data words, Address Control words, Idle Control words, and Payload Control words are illegal. Training is described in Section 8.8.

8.5 Data Words

Data words (DW[15:0]) are denoted by a logic low level on the associated xCTL signal. A common format is used in both the Transmit and Receive interfaces. Each Data word carries two bytes of address or payload data.

Three Data words have been defined; Address Data word (ADW), Payload Data word (PDW), and Training Data word (TDW). Each Address data word carries 16 bits of Port Address and allows systems to have Port Address ranges that are greater than that addressable using Payload Control words alone. A maximum of 8 ADWs can be concatenated in a sequence. A Payload Data word carries 2 bytes of payload data. Training Data words are used in conjunction with Training Control words to deskew the signals on the bus.

Table 8.8 below lists the relationship between various Control and Data words.

Table 8.8: Legal Word Ordering

Word	Legal Previous Word	Legal Next Word
ACW	ICW, PDW, TDW	ADW, PCW
PCW	ACW, ICW, ADW, PDW, TDW (when ACW always includes DIP-4) ACW, ADW (when ACW does not include DIP-4)	PDW
ICW	ICW, PDW, TDW	ACW, ICW, PCW, TCW (when ACW always includes DIP-4) ACW, ICW, TCW (when ACW does not include DIP-4)
TCW	ICW, TCW, TDW	TCW, TDW
ADW	ACW, ADW	ADW, PCW
PDW	PCW, PDW	ACW, ICW, PCW, PDW (when ACW always includes DIP-4) ACW, ICW, PDW (when ACW does not include DIP-4)
TDW	TCW, TDW	ACW, ICW, PCW, TCW, TDW (when ACW always includes DIP-4) ACW, ICW, TCW, TDW (when ACW does not include DIP-4)

8.6 DIP-4 Checksum

The DIP-4 field in a Control word covers the preceding sequence of Data words and the current Control word. The DIP-4 field in a Control word covers the preceding sequence of Data words and the current Control word, except that the DIP-4 may be omitted from the following Address Control word in certain cases (see DIP-4 description in Table 8.7). In the latter case, the DIP-4 field in the subsequent Payload Control word covers any sequence of Data words immediately preceding the Address Control word, the Address Control word itself, any Address Data words, and the current Payload Control word. An example of this is shown in Figure 8.7. In the presence of random errors, it offers the same error protection capability as conventional BIP-4 codes. But, it has the additional

advantage of spreading single-column errors (as might occur in a single defective line) across multiple parity bits. Appendix 13.1 discusses the error detection performance of this code.

Figure 8.6: DIP-4 Codewords for Address Control Words with DIP-4 Field

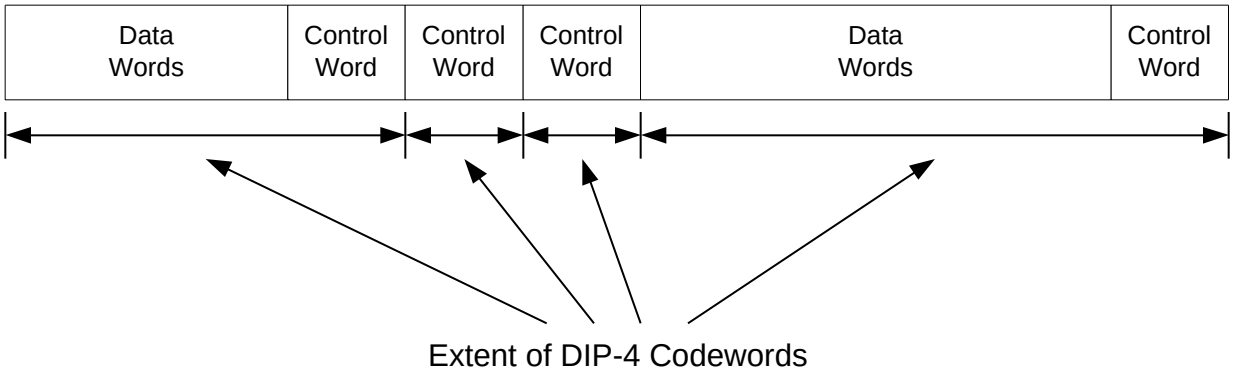
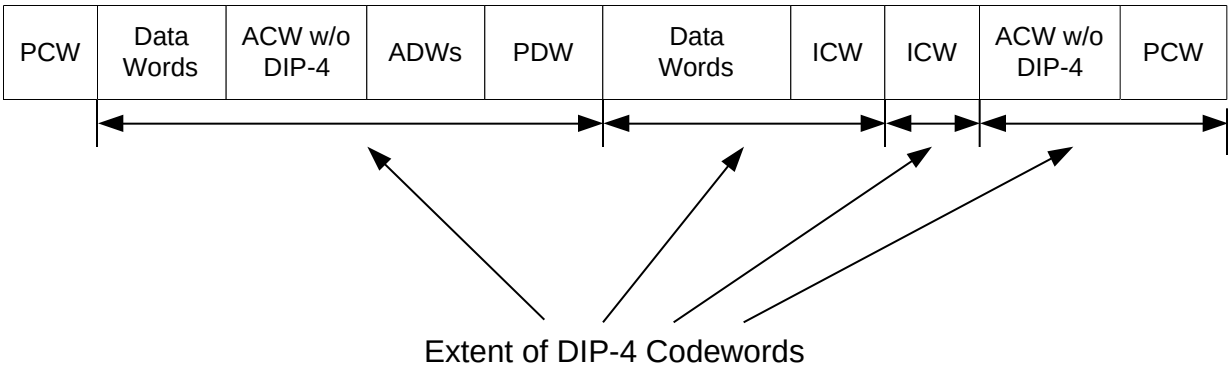
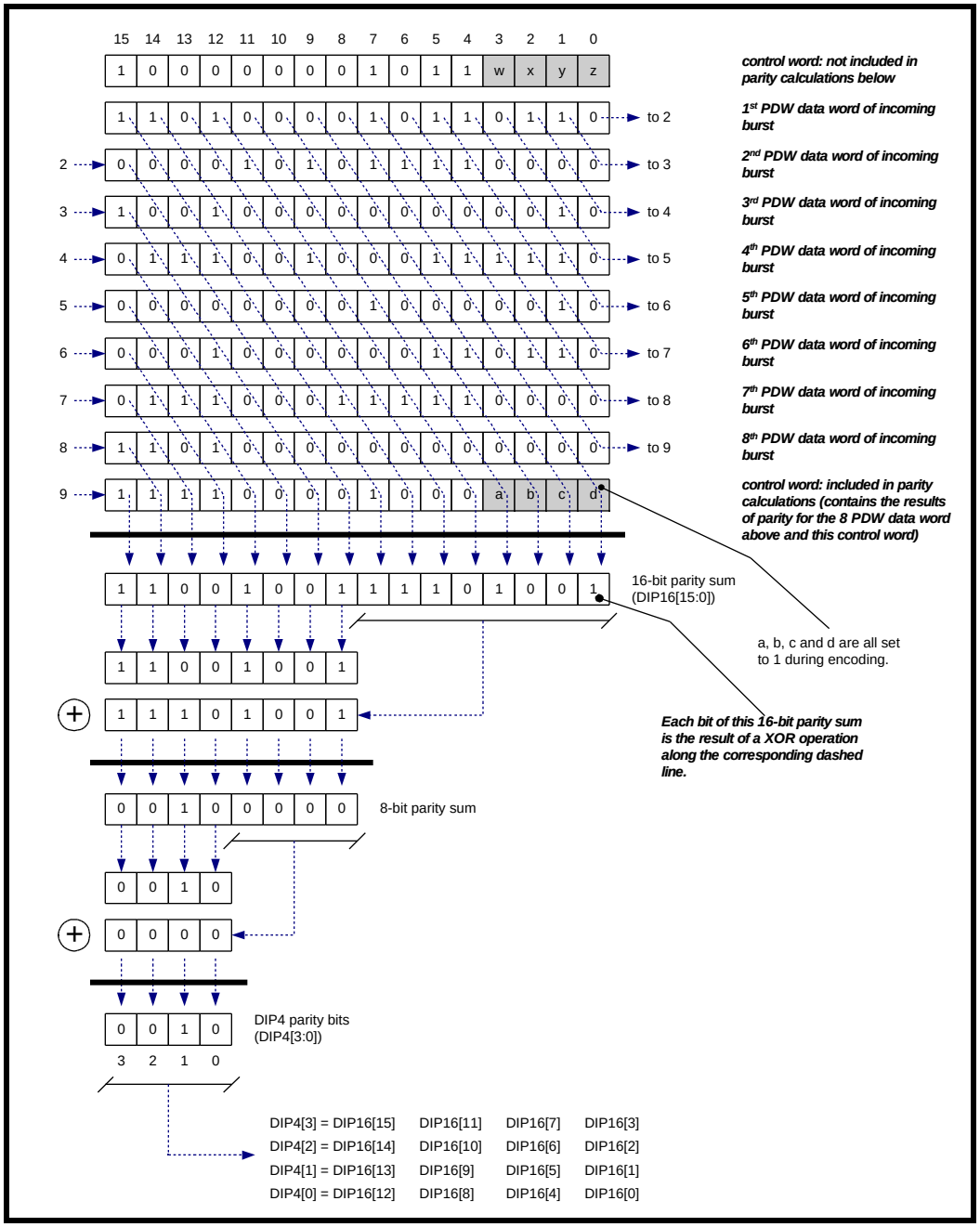


Figure 8.7: DIP-4 Codewords for Address Control Words without DIP-4 Field



A functional description of calculating the DIP-4 code is given as follows. A stream of 16-bit data words is shown in Figure 8.8. The MSB is at the leftmost column and time is moving downward. Thus, the first word received is at the top and the last word is at the bottom of the figure. The parity bits are generated by summing the data diagonally. In the Control word, the space occupied by the DIP-4 code (bits a,b,c,d) is set to all 1's during encoding. The final 16-bit checksum is split into two bytes, which are added to each other, modulo-2, to produce an 8-bit checksum. The 8-bit checksum is then further divided into two 4-bit nibbles, which are added to each other, modulo-2, to produce the final DIP-4 code. The procedure described applies to parity generation and to parity checking.

Figure 8.8: Example of DIP-4 Encoding (Odd Parity)

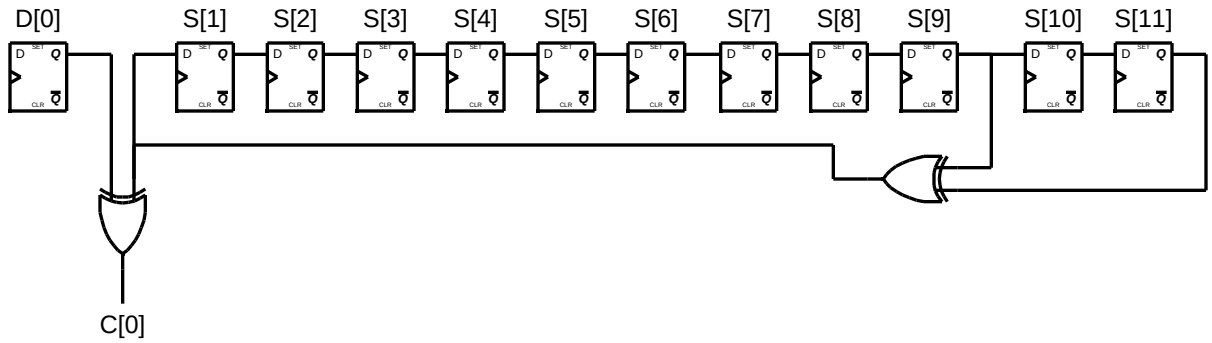


8.7 Scrambling

To control transition density, the signals in each Transmit and Receive interface of the SPI-5 bus are scrambled by a common $X^{11} + X^9 + 1$ LFSR stream cipher. Figure 8.9 shows a serial implementation of the scrambler operating on one of the signals in the source device of an interface. The scrambler advances by one

position every bit time. There is no phase relationship between the scramblers operation on different signals on the bus. The LSFRs of individual data channels and the control channel should be initialised with codes spaced out across the 2^{11}-1 codeset. A bit of the Control or Data word to be sent, prior to scrambling, is represented by the plaintext data "D[0]". The scrambled ciphertext data (C[0]) is placed on the data path.

Figure 8.9: Data Path Scrambler



To recover the data, each signal in the data path of the Transmit and Receive interface of the SPI-5 bus is de-scrambled by an independent $X^{11} + X^9 + 1$ LFSR stream cipher. The de-scrambler advances by one position every bit time. There is no phase relationship between the scramblers operation on different signals on the bus. The de-scrambler can be in the Locked state or in the Load state. When it is in the Locked state, the LFSR residue is synchronized to the LFSR in the scrambler. When the de-scrambler is in the Load state, it utilizes the regularly occurring Training sequence to synchronize to the LFSR in the scrambler. Appendix 13.2 provides an informative discussion on the Stream Cipher Scrambling Function.

8.8 Training Sequence

The data path of the SPI-5 bus operates at high speed and contains a number of dependent signals in each of the Transmit and Receive interfaces. It is, therefore, necessary to measure and to compensate for the delay variations between signals. A methodology involving periodic Training sequences is provided to address this requirement.

A data path Training pattern consists of 16 Training Control Words (TCW) and 16 Training Data Words (TDW). A Training sequence is a repeated sequence of Training patterns and may only originate from the Idle state (Figure 8.3). A Full Training sequence, consisting of at least TRAIN_LEN repetitions of the Training pattern, must be sent, at the first opportunity, if the last such sequence was sent longer than DATA_MAX_T word cycles ago. Sending Training sequences more

frequently with arbitrary repetition of the Training pattern is at the option of the data path source device. Data path training can be disabled by setting the bus parameter DATA_MAX_T to 0. TRAIN_LEN is a bus parameter and can have values from 1 to 8 when DATA_MAX_T > 0.

The sink device shall also force the sending of Training sequences via the Pool Status Channel when it has detected a data path loss-of-synchronization alarm. After sending the Training sequence, the data path returns to the normal operation. Table 8.9 shows a Training sequence for SPI-5. The ICW sent after the Training sequence is shown for illustrative purposes. If data is available to be transferred, an ACW or a PCW can be sent to start a Transfer.

Table 8.9: Data Path Training Sequence

Word cycle	*x*CTL[0]	Word	*x*DAT[15:0]	Description
0	1	ICW		ICW due to origination in the Idle control state. EOPS and DIP-4 fields reflect previous word cycles.
1 to 16	1	TCW	0FFF hex	First Training pattern, control
17 to 32	0	TDW	F000 hex	First Training pattern, data
—	—	—	—	Training patterns, control/data
32(N -1) + 1 to 32(N -1) + 16	1	TCW	0FFF hex	Last Training pattern, control For Full Training sequence, N ≥TRAIN_LEN
32(N -1) + 17 to 32(N -1) + 32	0	TDW	F000 hex	Last Training pattern, data
32(N -1) + 33	1	ICW	0000 hex	ICW shown; ACW, PCW, or TCW also legal

The Training sequence is designed to be robust in the face of large data delay variations over the signals constituting the Transmit and Receive interfaces. The underlying pattern in each bit channel is a 16 high / 16 low square wave. Assuming +/- 1 bit time of wander and +/- 3 bit times of delay variation in device and PC board delay, there will be at least 11 bit times during which a bus sink device can detect a Training Control word, prior to deskew. Larger variations would reduce the 11 bit times of margin on a one-for-one basis. The TDW is chosen to be the complement of the TCW and provides a distinct transition point to simplify delay variation measurements. After deskew the TCW / TDW transition will produce a synchronous wavefront across all bit channels in the bus

word. Note that the TCW / TDW encoding causes xDAT[15:12] to have the opposite phase from xCTL and xDAT[11:0].

8.9 Synchronization

The sink device attached to the Transmit or Receive interface monitors the synchronization state of the interface. Synchronization is lost when multiple DIP-4 codewords are in error. Synchronization may optionally be lost when multiple illegal/invalid words are detected. The loss-of-synchronization state will persist until a Training sequence has been received and synchronization is regained. The associated Pool Status Channel will be used to report the alarm (See Section 9.9). The source device of the affected Transmit or Receive interface will then respond by sending the Training sequence to re-align the bus and will cancel all credits previous granted by the Pool Status Channel.

9 POOL STATUS CHANNEL OPERATION

Each direction of the SPI-5 bus has its own independent Pool Status Channel. The Transmit Pool Status Channel (TSTAT) reports pool status in the PHY device back to the Link Layer device. The Receive Pool Status Channel (RSTAT) reports pool status in the Link Layer device back to the PHY device. The status channel implements a credit based flow control scheme, and is described in detail in this section.

For some applications, it may not be necessary or meaningful to convey status from the link layer device to the physical layer device on RSTAT. Therefore, Link Layer and PHY devices may omit the RSTAT input and associated signals. Other applications must implement the full interface.

9.1 Credit Pools

An SPI-5 data interface can support an enormous number of ports (see Section 8.2) but the bandwidth available for the status channel is limited (see Appendix 13.3). To accommodate a large number of ports on the status channel, multiple ports may be collected into what is called a Pool. Credits are granted and consumed on a per-Pool basis and all ports in a Pool share a common bank of credits. The pool number associated with a port address shall be uniquely identified by the lower order port address bit values PA[POOL_LEN-1:0]. Applications with a small number of ports may choose to have one port per pool, giving a simple one-to-one mapping between ports and credit pools. The “one port per pool” configuration shall be supported by all implementations down to a channelization of 64 ports.

Credits are granted by data path sink devices based on their ability to accept data. Each Pool Status report word grants credits to the data source device, allowing it to transfer data for the corresponding Pool across the interface. Each unit of credit permits a single 32-byte block of data to be sent to one of the ports sharing the Pool.

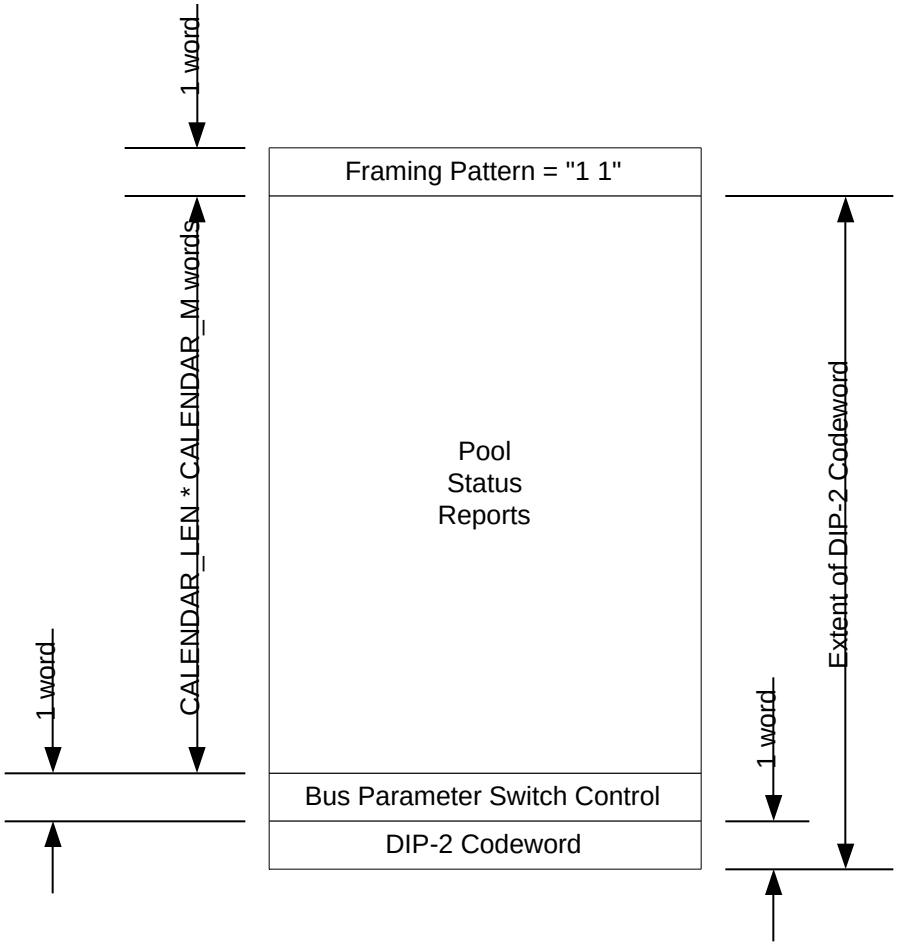
Credits are consumed by data path source devices whenever Payload Data words or Address Data words are sent. Credits are not consumed for Control words or for Training Data words. A credit is consumed for each 32-byte block of data that is transferred. A block containing less than 32 bytes consumes one whole credit. Thus, a partial block at the end of a packet consumes a whole credit even if it is shorter than 32 bytes in length. Similarly, a sequence of Address Data words at the beginning of a transfer always consumes a whole credit regardless of its length.

Credits are granted to Pools and are consumed for each block of data transferred from any port within the Pool. The source of a Transmit or Receive data interface only schedules transfers for ports whose associated Pool have sufficient credits to support the entire transfer.

9.2 Logical Status Frame Format

The Pool Status Channel message is organized in a logical frame structure, as shown in Figure 9.1. The elements of the status frame are 2-bit status words.

Figure 9.1: Pool Status Frame



A status frame begins with a Framing Word. The Framing Word is followed by a number of pool status report words, as defined by the Calendar (see Section 9.3). The pool status report words are followed by the Bus Parameter Switch Control Word (as described in Section 9.4). The frame ends with a DIP-2 odd parity checksum (as described in Section 9.5).

The length of a frame, in bits, is 6 overhead bits plus the number of Calendar entries times the Calendar repetition count times two bits per pool status report word. The 6 bits for overhead consist of the Framing Word (2), the Bus Parameter Switch Control Word (2), and the DIP-2 checksum (2).

The two-bit Framing Word of “11” is reserved to delimit the beginning/end of a status frame. The three other two-bit patterns (“01”, “10” and “00”) are used for the body of the status frame to indicate the fill level of each pool, and are placed on the channel in a time-division-multiplexed manner in accordance with the Calendar.

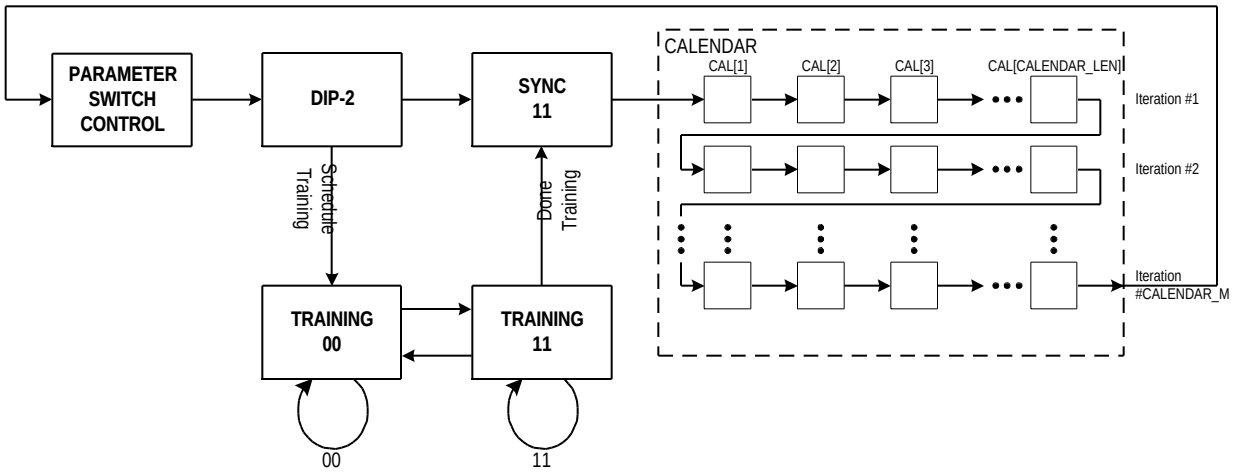
A framer in the sink device of a Pool Status Channel monitors the received pool status frame for the framing pattern. When the framer is in-frame, the pool status report words are used to update the credits of the Pools. The framer monitors for framing pattern errors, DIP-2 errors and optionally Bus Parameter Switch Control Word errors. Occurrence of multiple framing errors places the framer in the out-of-frame state. When the framer is in the out-of-frame state, all previously granted credits are canceled and remain so until the framer returns to the in-frame state. The framer returns to the in-frame state when it observes multiple consecutive error-free pool status frames. Although no specific number of framing errors is specified, it is a requirement that any single error event shall not cause loss of synchronization.

Note that the pattern “11” is a valid DIP-2 codeword and is also part of the Training pattern (see Section 9.7). The framer must not lock onto these occurrences as frame boundaries.

9.3 Pool Status Calendar

The order of pool status report words in a frame is defined by a Calendar structure, as shown in Figure 9.2. The number of entries in the Calendar is given by the bus parameter CALENDAR_LEN. The number of times that the calendar is traversed for each frame is given by the bus parameter CALENDAR_M. Entries in the calendar are traversed sequentially.

Figure 9.2: Pool Status Calendar



Each Calendar entry has an associated pool address (CAL[i]), which may optionally be provisionable. (For example, in the case of a single channel device, CAL[i] need not be provisionable.) As the Calendar is traversed, the status of the Pool associated with each Calendar entry is placed on the Pool Status Channel. The status is encoded using one of the three possible 2-bit Pool status report word values defined in Table 9.1.

Table 9.1: Pool Status Report Format

***x*STAT[1]**	***x*STAT[0]**	Description
1	1	In-band Framing Pattern The framing pattern is sent once per pool status frame in normal operation. This status value is also repeated as part of the training pattern.
1	0	SATISFIED The SATISFIED status indicates that the FIFO of the corresponding sink Pool is almost full. When the SATISFIED status is received, no further credits are granted. Credits granted previously by HUNGRY or STARVING status reports remain available.

***x*STAT[1]**	***x*STAT[0]**	Description
0	1	HUNGRY The HUNGRY status indicates that the FIFO of the corresponding sink Pool is partially empty. When a HUNGRY status is received, the amount of credits at the source Pool is increased to MAXBURST2 blocks if the current value is less than MAXBURST2. If the amount of credit remaining is currently greater than MAXBURST2, due to a previous STARVING status report, the credit count is left unchanged.
0	0	STARVING The STARVING status indicates that the FIFO of the corresponding sink Pool is almost empty. When a STARVING status is received, the amount of credits at the source Pool is set to MAXBURST1 blocks. This status value is also repeated as part of the training pattern.

Note that a pool address may appear more than once in a calendar. This is helpful in situations where the pools have unequal bandwidths. For example, a high bandwidth pool can have a proportionally greater share of entries in the calendar to allow more frequent status updates.

Not that if a calendar entry is unpopulated or unprovisioned it should be sent with xSTAT=10 “Satisfied”.

Because the Transmit and Receive interfaces operate independently, their calendars may be provisioned differently. However, it is a requirement that the source and sink devices of a given interface have identical Calendar configurations in order to operate properly. The method for making changes to the calendar configuration, while also ensuring that both sides have identical settings is discussed in the next section. A discussion on performance considerations for the Pool Status Channel is given in Appendix 13.3.

9.4 Bus Parameter Switch Control Word (BPSCW)

Transmit and Receive interfaces may contain two sets of bus parameters (e.g., CAL[i], MAXBURST1[i]). At any time, one set is active and one set is standby. The standby set of parameters can be updated with new information without affecting normal operation. The Bus Parameter Switch Control Word (BPSCW)

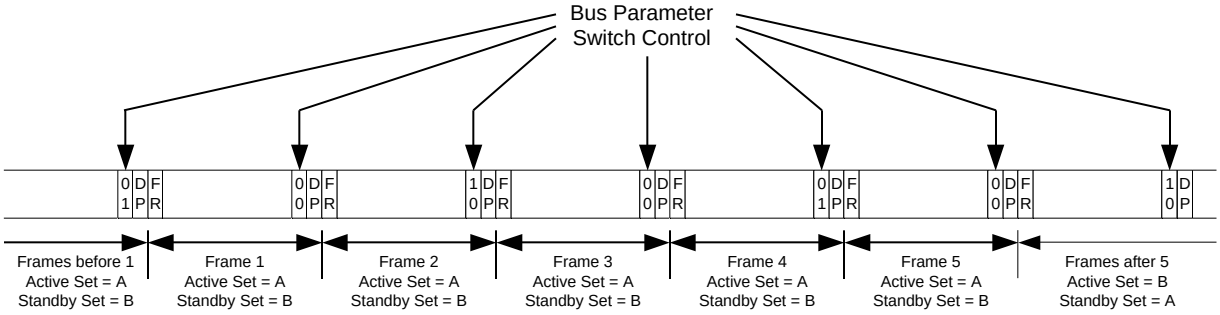
indicates which parameter set is active, and robustly indicates when switching from active to standby.

During steady-state operation (i.e. when parameter sets are not being changed), the BPSCW indicates which parameter set is active. When the BPSCW is “01”, the first parameter set is active and the second set is standby. When the BPSCW is “10” the second parameter set is active and the first set is standby.

To change parameter sets from active to standby, special five-word patterns are transmitted in the BPSCW of five consecutive Pool Status Frames. After receiving the final word of the pattern in the fifth frame, the pool status receiver switches to the standby parameter set.

When switching from parameter set “01” to “10,” the pattern of five BPSCWs is “00”, “10”, “00”, “01”, “00”. When switching from parameter set “10” to “01” the pattern of five BPSCWs is “00”, “01”, “00”, “10”, “00”. These patterns are robust in the sense that any two-bit error can be corrected, and any three-bit error can be detected. A status channel receiver that is capable of switching its bus parameters shall be capable of correcting double bit errors in the switching pattern. Figure 9.3 shows the sequencing of a parameter switch.

Figure 9.3: Bus Parameter Switch Timing



To facilitate recovery from error or startup conditions, a status channel receiver searches for five or more consecutive identical BPSCWs (either “10” or “01”). When it detects such a condition, it shall switch to the appropriate parameter set. A status channel receiver may optionally report such switches, as well as errors in the BPSCW sequence.

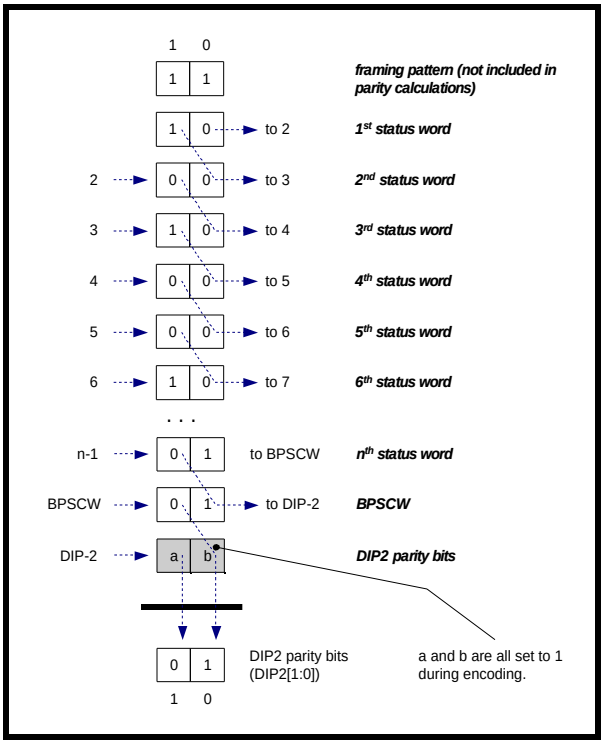
9.5 Serialization for Transmission

The two-bit Pool status report words from the Logical Pool Status Frame are transmitted on TSTAT/RSTAT. Words are transmitted MSB (xSTAT[1]) first.

9.6 DIP-2 Checksum

A DIP-2 odd parity checksum is sent at the end of each status frame. The DIP-2 code is computed diagonally over the bus parameter control field and all preceding Pool status report words sent after the last Framing Pattern, as shown in Figure 9.4. The first word is at the top of the figure and the last word is at the bottom of the figure. The parity bits are computed by summing diagonally. Bits a and b correspond to the space occupied by the DIP-2 parity bits, which are set to 1 during encoding. The in-band framing pattern is not included in the parity calculation. The procedure described applies to both parity generation and to parity checking functions.

Figure 9.4: Example of DIP-2 Encoding (Odd Parity)



An error monitor in the sink device continuously verifies the DIP-2 codewords received. After multiple DIP-2 errors, the monitor enters the DIP-alarm state and all previously granted credits to Pools are cancelled and will remain cancelled. After multiple consecutive error-free codewords are received, the monitor returns to the DIP-normal state and credits can be accumulated afresh. Although no specific number of DIP-2 errors is specified, it is a requirement that any single error event shall not cause an alarm.

Note that a receiver should not wait for the DIP-2 validation of the contents of a status frame before acting on the status reports contained in the status frame.

9.7 Scrambling

The Pool Status Channel signals TSTAT/RSTAT are scrambled as described in Section 8.7.

9.8 Training Sequence

A Pool Status Channel receiver demultiplexes the received bit stream and forms 2-bit pool status report words before it can attempt to frame on it. A Training Sequence is occasionally inserted into the transmitted bitstream to allow the demultiplexer to properly synchronize with the 2-bit pool status report words.

The training pattern consists of 8 words of “00” followed by 8 words of “11”. The training sequence consists of the training pattern repeated at least TRAIN_LEN times.

The training sequence is inserted between two consecutive Logical Pool Status frames, each time that a provisionable timer expires. It is inserted between the DIP-2 checksum at the end of the status frame in which the timer expired, and the Framing Word “11” of the following status frame. The timer counts in units of 2-bit status words and is provisioned with the bus parameter FIFO_MAX_T. Status channel training can be disabled by setting the bus parameter FIFO_MAX_T to 0. TRAIN_LEN is a bus parameter and can have values from 1 to 8 when FIFO_MAX_T > 0.

9.9 Loss of Data Synchronization (LODS) Alarm

An SPI-5 data path sink continuously monitors the data path for proper synchronization and error free operation. When a data sink detects multiple errors on the data path, it declares a Loss Of Data Synchronization (LODS) condition. This alarm condition is reported by sending an LODS alarm pattern back to the data path source on the Pool Status Channel. Although no specific number of data path errors is specified, it is a requirement that any single error event shall not cause loss of synchronization.

The LODS alarm status pattern is sent immediately when an LODS condition is detected, without regard to the current position in the status frame. The pattern is repeated continuously until the data path has regained synchronization and has received multiple error-free control words.

The LODS alarm status pattern consists of the training pattern (as described in Section 9.7) sent at least 16 times. Since the LODS pattern uses the “11” bit pattern that is reserved as the Pool Status Channel framing word, continuous LODS patterns will force the Pool Status Channel receiver into the out-of-frame state. Receipt of no more than 12 repetitions of the LODS pattern shall be sufficient to force the receiver to the out-of-frame state. Consequently, the data path source cancels all previously granted credits. Credits are only refreshed after the Pool Status Channel returns to the in-frame state.

10 BUS PARAMETERS

The SPI-5 bus is configured by five types of parameters. The first configures the Pool status channel, the second data path credits granted by Pool status reports, the third the Burst Admission Procedure, the fourth the range of ports addressable, and the last the Training sequences for the data path and the Pool status channel.

The complete list of bus parameters is listed below in Table 10.1. The column labeled “P” denotes whether the associated parameter is configurable within devices that are connected to the bus. A device is deemed compliant if it supports output interface configuration on parameters labeled “Y/O”. The column labeled “CP” describes whether the parameter is configurable on a per-Pool basis or only configurable on a per-interface basis. A device is deemed compliant if it supports either per-Pool or per -Interface configuration on parameters labeled “P/I”.

Table 10.1: Bus Parameters

Parameter	Definition	P	CP	Unit
CAL[i]	Credit pool reported at Calendar location i	O	I	Pool number
CALENDAR_LEN	Length of the Calendar sequence. CALENDAR_LEN = 1, 2, 4, or 8*n where n is a positive non-zero integer	O	I	Entries
CALENDAR_M	Number of traversals of the Calendar within a Pool status frame. CALENDAR_LEN * CALENDAR_M =8*m, Where m is an interger greater than 1	Y	I	Repetitions
MAX_CALENDAR_LEN	Maximum supported value of Calendar entries	N	I	Entries
MAXBURST1	Number of credits, in 32 byte blocks, granted when Pool status channel indicates Starving	Y	P/I	32 byte blocks
MAXBURST2	Minimum number of 32 byte blocks a FIFO can accept when Pool status channel indicates Hungry. MAXBURST2 ≤ MAXBURST1	Y	P/I	32 byte blocks

Parameter	Definition	P	CP	Unit
TOKEN_MAX	Maximum number of tokens for 32 byte blocks that the input is allowed to accumulate. TOKEN_MAX ≥ **min(($2N_p$ - 1), 7)**, where N_p is the number of Credit Pools	Y/ O	I	32 byte blocks
TOKEN_GEN	Number of Word Cycles that must expire in order to accumulate another token. TOKEN_GEN ≤ 12	Y/ O	I	Word Cycles
PADDR_LEN	Number of bytes in the Physical Address field of the port address. Minimum number of address bytes explicitly sent and uniquely decoded at the start of every Transfer.	Y	I	Bytes
TOP_PADDR	Highest Physical Address supported. The range of supported Physical Addresses is 0 to TOP_PADDR.	N	I	Ports
POOL_LEN	Number of bits in the port address used to uniquely decode for the associated credit Pool	O	I	Bits
TRAIN_LEN	Minimum number of repetitions of the Training pattern in the mandatory training sequence required during the DATA_MAX_T and the FIFO_ MAX_T intervals. TRAIN_LEN ≤ 8	Y	I	Repetitions
DATA_MAX_T	Nominal interval between training sequences on Data Path interface.	Y	I	Word cycles
FIFO_MAX_T	Nominal interval between training sequences on Pool Status interface.	Y	I	Word cycles
RSTAT_PRESENT	Indicates presence of Pool Status Channel	Y	I	Boolean

Notes:
P : Provisionable Parameter –

Yes [Y], No (Fixed within device) [N], Optional [O],
Yes (output) Optional (input) [Y/O]

CP : Credit Pool Configuration –
per-Pool [P], per-Interface [I], either [P/I]

Devices communicating over a SPI-5 bus may optionally support multiple sets of bus parameters. A common application would be the support for the provisioning and the un-provisioning of ports. The Calendar may have to be modified to include the enabled ports and to exclude the disabled ports. A mechanism is provided within the Pool Status Channel to synchronize the switching from one parameter set to the next.

RSTAT_PRESENT is only present in data-path source devices. It indicates the presence, or not, of an incoming pool status channel. It is provisioned based on the capabability of the partnering data-path sink device. If that device provides a pool status channel (and it is connected) this parameter should be set to true. If that device does not provide a pool status channel (or it is not connected) this parameter should be set to false. If RSTAT_PRESENT=true pool credit based flow control is enabled in the data-path source device. If RSTAT_PRESENT=false, data-path source transmissions are only limited by the burst admission procedure.

10.1 Calendar Parameters

The sequence of ports reported in status frame of a Pool status channel is defined in a data structure called Calendar, where CAL[i], i = 1,…,CALENDAR_LEN, refers to the i^{th} credit Pool in the repeating sequence. CALENDAR_LEN must be at least as large as the number of active Pools in the system. Certain applications may wish to devote more than one Calendar entry to a Pool. An example would be to scale the number of entries in proportion to the sum of the bandwidth of ports sharing the Pool. Within a status frame, the Calendar sequence (of length CALENDAR_LEN) is traversed CALENDAR_M times. The minimum product of CALENDAR_LEN times CALENDAR_M is 16 to facilitate parallel processing of the calendar.

Examples:

1. Single OC-768 port:
   CALENDAR_LEN = 1, CALENDAR_M = 16, CAL [1] = 1.
2. Four OC-192 or 10 Gb/s Ethernet ports:
   CALENDAR_LEN = 4, CALENDAR_M = 4, CAL[i] = 1, 2, 3, 4.
3. Two OC-192 channels (ports 1 and 2), eight OC-48 channels (ports 3 through 10):
   CALENDAR_LEN = 16, CALENDAR_M = 1,
   CAL[i] = 1, 2, 3, 4, 1, 2, 5, 6, 1, 2, 7, 8, 1, 2, 9, 10

The values of CALENDAR_LEN, CALENDAR_M and CAL [i] entries must be identical in both the PHY and Link Layer devices for each interface. They need not be identical on both the Transmit and Receive interfaces. The maximum supported value of CALENDAR_LEN is contained in the parameter MAX_CALENDAR_LEN, which is device-specific. MAX_CALENDAR_LEN need not be identical on either side of the Transmit or Receive interface. Applications must ensure that the value of CALENDAR_LEN must not exceed the lesser of the MAX_CALENDAR_LEN parameters at the source and sink sides of a Pool status channel.

10.2 Other Parameters

The MAXBURST1 and MAXBURST2 parameters may be configured to apply globally to all Pools in an interface, or to apply on a per-Pool basis. In either case, both parameters must be consistently configured at the PHY and Link Layer devices for each interface. They do not need to be identical between the Transmit and Receive interfaces. MAXBURST1 must be greater than or equal to the associated MAXBURST2.

The PADDR_LEN parameter is configured on a per-interface basis. Both the PHY and Link Layer devices for each interface must have a consistent view of this parameter. Every payload Transfer shall have the number of Address Control words and Address Data words specified by PADDR_LEN.

The POOL_LEN parameter is configured on a per-interface basis. Both the PHY and Link Layer devices for each interface must have a consistent view of this parameter.

For the data path and the Pool status channel deskew processes, configuration of TRAIN_LEN, DATA_MAX_T and FIFO_MAX_T is mandatory on the source side of the interface only. Corresponding parameters can optionally be configured on the sink side of the interface to timeout the synchronization state, especially for test purposes (see MAXLOCK, MINLOCK in Appendix 13.2.3).

11 BUS START-UP

When disabled, the data path sink is placed in the Loss of Data Synchronization (LODS) state and empties all its FIFOs. Any outstanding credits in the data path source are cleared. The Pool status channel is placed in the Disabled state and will send the loss-of-synchronization (LODS) pattern continuously. Thus, the data path source shall send continuous Training patterns. The bus parameters can now be configured. At the same time, the data path sink monitors for the Training sequence and synchronizes to signals on the interface. The Pool status channel will begin sending valid status when it is enabled and the associated data path sink has exited the LODS state. The data path sink shall ignore all incoming data until it has acquired synchronization with the data and confirmed by receipt of at least one more Training pattern.

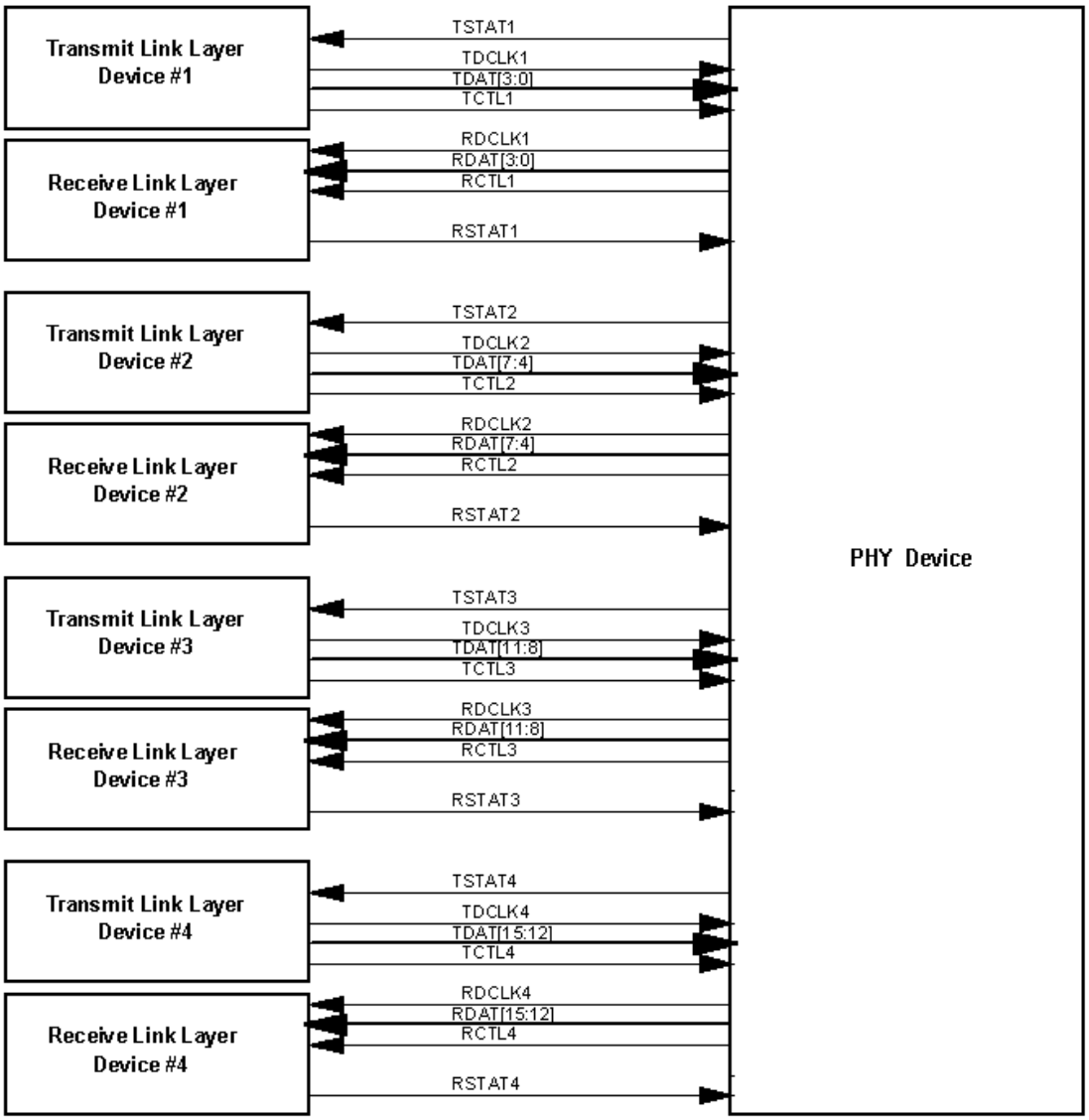
In the case where the data path sink is disabled but the data path source remains active, events at the sink follow the same behavior as above. The Pool status channel will send continuous LODS patterns which causes the data path source to clear all previously granted credits and to start sending the Training sequence. Thus, it behaves, substantially, as though it too had been disabled.

In the case where the data path source is disabled but the data path sink remains active, events at the source follow the same behavior as above. The source will clear all its credits. The sink will enter the LODS state. From this point forward, the behavior of both the data path source and sink is, substantially, similar to that of simultaneous disabled to both ends of an interface.

12 APPENDIX: NARROW BUS INTERFACE MODE

This section describes the recommended implementation for applications that support four streams with OC-192 capacity via a narrow bus mode of operation. This implementation can also be used to support four OC-192 PHY devices from a single Link Layer device. SPI-5 interface I/Os can optionally be reconfigured by splitting and extending the bus as shown in the block diagram of Figure 12.1.

Figure 12.1: SPI-5 Interface, 4x4bit Version


Transmit Link Layer Device #1
Receive Link Layer Device #1
Transmit Link Layer Device #2
Receive Link Layer Device #2
Transmit Link Layer Device #3
Receive Link Layer Device #3
Transmit Link Layer Device #4
Receive Link Layer Device #4
PHY Device
TSTAT1
TDCLK1
TDAT[3:0]
TCTL1
RDCLK1
RDAT[3:0]
RCTL1
RSTAT1
TSTAT2
TDCLK2
TDAT[7:4]
TCTL2
RDCLK2
RDAT[7:4]
RCTL2
RSTAT2
TSTAT3
TDCLK3
TDAT[11:8]
TCTL3
RDCLK3
RDAT[11:8]
RCTL3
RSTAT3
TSTAT4
TDCLK4
TDAT[15:12]
TCTL4
RDCLK4
RDAT[15:12]
RCTL4
RSTAT4

The transfer of data is done in 16-bit words as in the single-interface case. The 16-bit words are divided into 4-bit nibbles and sent across the interface in a 4-bit serial mode. I.e., for interface #1, bits[15:12] are sent as a serial string via TDAT[3]/RDAT[3], bits[11:8] via TDAT[2]/RDAT[2], bits[7:4] via TDAT[1]/RDAT[1], and bits[3:0] via TDAT[0]/RDAT[0] . Table 12.1 shows the sequence of 16-word bits at the interface per nibble cycle. The example shows a Data word (xDW) transfer followed by a Control word (xCW) transfer on interface #1. The Transmit Control and Receive Control signals stay high during all 4 nibble cycles when a control word is sent, and remain low during all 4 nibble cycles when a data word is sent.

Table 12.1: Data path sequence for interface #1

Nibble cycle (TDCLK1/RDCLK1)	TCTL1 / RCTL1	TDAT[3] / RDAT[3]	TDAT[2] / RDAT[2]	TDAT[1] / RDAT[1]	TDAT[0] / RDAT[0]
1	0	xDW[15]	xDW[11]	xDW[7]	xDW[3]
2	0	xDW[14]	xDW[10]	xDW[6]	xDW[2]
3	0	xDW[13]	xDW[9]	xDW[5]	xDW[1]
4	0	xDW[12]	xDW[8]	xDW[4]	xDW[0]
5	1	xCW[15]	xCW[11]	xCW[7]	xCW[3]
6	1	xCW[14]	xCW[10]	xCW[6]	xCW[2]
7	1	xCW[13]	xCW[9]	xCW[5]	xCW[1]
8	1	xCW[12]	xCW[8]	xCW[4]	xCW[0]

The clocks TDCLK1..4 and RDCLK1..4 are locked to the respective data and control signals of each interface, but can be asynchronous between each other. The minimum clock frequency is 622 MHz.

Each status channel signal transfers 1-bit serial pool status indication for the respective interface.

The protocol for this narrow bus interface mode is exactly the same as for the 16-bit wide interface. This means that the control word coding, the DIP-4 calculation, status indication and flow control follow the same rules as for the wide interface, always with respect to 16-bit words. Each bit line is scrambled and descrambled by a $X^{11} + X^9 + 1$ LFSR as shown in Figure 8.9 and Appendix 13.2. The descrambler LFSR is automatically synchronized with the training pattern, as is the case for the 16-bit wide interface.

As a slight deviation from the 16-bit interface mode, the Training Pattern of the narrow interface consists of 4 Training Control words (TCW) and 4 Training Data

words (TDW) only. On each individual data bit line, this results in the same number of 16 repetitions of bit patterns as in the case of the 16-bit interface. A Training Sequence is a repeated sequence of Training Patterns and can only originate from the Idle state (see Figure 8.4). TRAIN_LEN is the minimum number of repetitions of the Training Pattern in a full Training Sequence. Training Control and Data words are identical to those defined for the 16-bit wide interface.

13 INFORMATIVE APPENDICES

13.1 Appendix: Error Detection Capability of the DIP-4 Code

In this section, the undetected error probability of the DIP-4 code is evaluated for a burst data transfer length of 64 bytes, in the presence of random bit errors.

With the DIP-4 code, each constituent parity bit is computed over 66*8/4-1 information bits. These information bits, together with the parity bit, constitute a single-parity codeword. In the assumed case of odd parity, the value of the parity bit is set such that the total number of ones in the codeword is an odd number. It can easily be seen that any odd number of errors can be detected in a single-parity codeword. Hence, an undetected error event for a given codeword occurs whenever an even number of errors falls on that codeword. An undetected error event on a given burst transfer occurs when an undetected error event occurs on at least one of the constituent codewords.

Let P_{UC} and P_{UT} denote, respectively, the undetected error probabilities in a single-parity codeword and in a DIP-4 burst transfer. Let L denote the number of bits in a codeword. Let *p* denote the probability of a bit error on the electrical interface. Enumerating all combinations of even-numbered errors in the codeword, we have

$$P_{UC} = \sum_{\substack{i=2,\\ i\text{even}}}^{L} \binom{L}{i} p^i (1-p)^{L-i}.$$

Since an undetected error in a burst transfer corresponds to an undetected error in at least one codeword,

$$P_{UT} = 1 - (1 - P_{UC})^4.$$

Table 13.1 shows the corresponding probabilities of undetected error in a burst transfer, for the DIP-4 code, over a range of bit error rates. Also shown for comparison are corresponding probabilities without an error-detection code. It can be seen that a DIP-4 code reduces the undetected error probability by several orders of magnitude. While the bit error rates in well-designed implementations may already be very low, the DIP-4 code can reduce the undetected error probability to extremely negligible levels with minimal added complexity to the interface implementation. Note however that longer burst transfers increase the DIP-4 extent, which increases the probability of an undetected error.

Table 13.1: Error Detection Capability of DIP-4 Code

Bit Error Rate	Probability of Undetected Error	
	Without Error Detection	With DIP-4 Code
1.0E-6	5.28E-4	3.458E-8
1.0E-7	5.28E-5	3.458E-10
1.0E-8	5.28E-6	3.459E-12
1.0E-9	5.28E-7	3.464E-14
1.0E-10	5.28E-8	3.441E-16
1.0E-11	5.28E-9	(tiny)

13.2 Appendix: Stream Cipher Scrambling Function

The purposes of scrambling are to:

- Produce a more uniform distribution of run lengths in otherwise unencoded data to aid in bit timing recovery.
- Spread the transmission spectrum to minimize electromagnetic compatibility problems.

The operation of the scrambling function is intended to be transparent to the other aspects of the SPI-5 protocol, including:

- The sequence of control and data words in the data path.
- The sequence of status words in the status path.
- The training pattern in the data and status paths.
- Error detection and recovery mechanisms in the data and status paths.

In this context transparency means that the other aspects of the SPI-5 protocol are not affected by the behavior of the scrambling function; however, the behavior of other aspects of the SPI-5 protocol may affect the scrambling function. The effects of other aspects of the SPI-5 protocol on the scrambling function can be summarized as:

- Off - No signal is being transmitted.

- Training - Continuous training patterns are being transmitted. The duration of training may range from a few repetitions of the training pattern, to milliseconds, to many seconds.

- Active – Normal data is being transmitted. The training pattern may be periodically interspersed with the normal data.

To be transparent to the other aspects of the SPI-5 protocol, the descrambler must be able to:

- Acquire synchronization during the training phase.

- Maintain synchronization during the active phase.

- Respond to potential loss of synchronization and attempt to resynchronize when the training pattern is received.

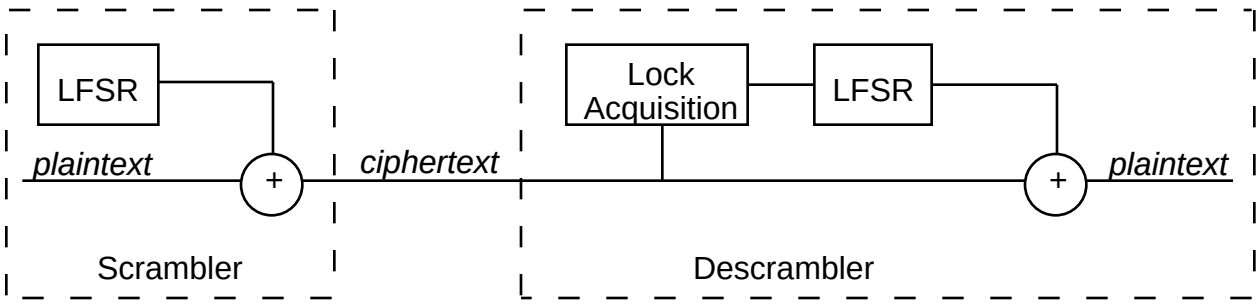
The SPI-5 protocol layers detect loss of synchronization in both the data path and the status path. Loss of synchronization status can be provided to the descrambler function. Alternatively the descrambler function itself can detect loss of synchronization by timing the intervals between detection of the training pattern in normal operation. The descrambler example uses a timer. However both methods are permitted and equally valid. In a parallel data path a common loss of synchronization detector will require considerably less hardware, at the expense of communicating across the individual signal channels.

The remainder of this annex explains the theory of stream cipher coding, followed by examples of stream cipher scrambler and descrambler designs that conform to the requirements for SPI-5. These examples have been chosen to illustrate design techniques. They are not intended to define either a recommended or an optimal design. Other conforming designs have been developed and are not excluded by this specification.

13.2.1 Stream Cipher Principles

The stream cipher method of scrambling is used to control run length while reducing electromagnetic emissions in the otherwise unencoded serial bit streams. Stream ciphering randomizes the data spectrum by the addition of a pseudorandom key sequence to the plaintext sequence transmitted by the SPI-5 protocol layers. The pseudorandom sequence is subtracted by the receiver in order to recover the transmitted data. Stream ciphering also has the desirable property that single bit errors in the received ciphertext bit stream decode as single bit errors in the recovered plaintext stream, with no transposed or duplicated error bits. Figure 13.1 illustrates these concepts.

Figure 13.1: Stream Cipher Architecture



For transmission, plaintext is scrambled by the modulo 2 addition of a pseudorandom sequence with the plaintext data to be transmitted. The pseudorandom sequence is generated by a Linear Feedback Shift Register (LFSR), and is exclusive ORed with the plaintext data stream to produce the transmitted ciphertext. As noted above, the sequence length is chosen to control run length and reduce electromagnetic emissions. A sequence length of 11 bits has been shown to provide sufficient run length and emissions reduction with a reasonable implementation complexity.

At the receiver, the plaintext is recovered by modulo 2 subtraction of the identical pseudorandom sequence from the ciphertext. This can only occur when the receiver's LFSR is synchronized with the transmitter's LFSR, so that the pseudorandom sequence applied to decode the ciphertext is the same sequence applied to encode the plaintext.

Synchronization of the descrambler can be achieved by using the SPI-5 training pattern. Scrambling the training pattern produces patterns in the ciphertext which the descrambler uses to recognize the receipt of the training pattern. The descrambler uses this information to hypothesize a value for the LFSR's seed, and check the resulting plaintext output to determine if synchronization of the LFSRs has been achieved.

Appendices 13.2.2 and 13.2.3 show examples of stream cipher implementation for encoders and decoders respectively. The designs are described as Verilog modules.

13.2.2 Stream Cipher Scrambler Example

```
//
// STREAM CIPHER SCRAMBLER
//
// for bit serial data path
//
// txclk:      transmit bit clock
// txen:    transmit enable (disables scrambler)
// tnrzdin: unscrambled input NRZ data stream (plaintext)
// tnrzdout:   scrambled output NRZ data stream (ciphertext)
//
module scrambler (txclk, txen, tnrzdin, tnrzdout);
input txclk, txen, tnrzdin;
output tnrzdout;
wire [11:0] s;    // scrambler key stream
reg [10:0] rs;    // key stream register
reg tnrzdout;     // ciphertext output bit
//
// KEY STREAM
//
assign s[11:1] = rs[10:0];    // shift previous bits
assign s[0] = s[11] ^ s[9];      // generate newest bit
always @(posedge txclk)
      if ( !txen )
         rs[10:0] = #1 11'h0;    // reset key stream
      else if ( rs == 11'h0 )
         rs[10:0] = #1 11'h1;    // initialize key stream to non-zero
      else
         rs[10:0] = #1 s[10:0];     // save current key stream
//
// CIPHERTEXT STREAM
//
always @(posedge txclk)
      tnrzdout = #1 s[0] ^ tnrzdin;    // scramble NRZ data bit
endmodule
```

13.2.3 Stream Cipher Descrambler Example

```
//
// STREAM CIPHER DESCRAMBLER
//
// for bit serial data path
//
// rxclk:      recovered bit clock
// rxsd:    signal detect (disables descrambler)
// rxcd:    clock detect status (disables descrambler)
// rnrzdin: scrambled input NRZ data stream (ciphertext)
// rnrzdout:   descrambled output NRZ data stream (plaintext)
// invert      invert training pattern for DAT[15:12]
// testmode:   enable test mode with short timeout
//
module descrambler (rxclk, rxsd, rxcd, rnrzdin, rnrzdout, invert,
testmode);
input rxclk, rxsd, rxcd, rnrzdin, invert, testmode;
output rnrzdout;
wire [11:0] c;    // ciphertext input stream
wire [21:0] h;    // descrambled hypothesis stream
reg [20:0] rh;    // hypothesis stream register
wire [31:0] tp;      // training pattern in plaintext stream
wire [31:0] hp;      // training pattern in hypothesis stream
wire enable;      // enable descrambler
wire load;     // load descrambler from ciphertext stream
wire tpat;     // valid training pattern received
wire [11:0] u;    // descrambler key stream
reg [10:0] ru;    // key stream register
reg rnrzdout;     // plaintext output bit
reg [16:0] locktime; // lock timer - time since training pattern
received
reg locked;    // 1 = locked/synchronized, 0 = not synchronized
//
// CIPHERTEXT STREAM
//
assign c[11:1] = ru[10:0];    // shift previous bits
assign c[0] = invert ^ rnrzdin;     // get newest bit
//
// HYPOTHESIS STREAM
//
assign h[21:1] = rh[20:0];    // shift previous bits
assign h[0] = c[11] ^ c[9] ^ c[0];  // generate newest bit
always @(posedge rxclk)
      rh[20:0] = #1 h[20:0];     // save current bits
//
// HYPOTHESIS PATTERN RECOGNITION
```

```
//
assign tp = { 16'hFFFF, 16'h0000 };    // plaintext training pattern
assign hp = tp ^ { 9'h0, 2'b11, 14'h0, 2'b11, 5'h0 };
      // convolved training pattern
assign enable = ( rxsd & rxcd );
      // enable if signal detect and clock detect
assign load = ( !locked ) && ( h[20:0] !== hp[20:0] );
assign tpat = ( locked ) ? ( h[20:0] == tp[20:0] ) :
                           ( h[20:0] == hp[20:0] );
//
// KEY STREAM
//
assign u[11:1] = ru[10:0];    // shift previous bits
assign u[0] = u[11] ^ u[9];      // generate newest bit
always @(posedge rxclk)
      if ( !enable )
         ru[10:0] = #1 11'h0;    // reset key stream
      else if ( load )
         ru[10:0] = #1 c[10:0];     // load new key stream
      else
         ru[10:0] = #1 u[10:0];     // save current key stream
//
// PLAINTEXT STREAM
//
always @(posedge rxclk)
      rnrzdout = #1 u[0] ^ rnrzdin;    // descramble NRZ data bit
//
// SYNCHRONIZATION STATE
//
`define MAXLOCK (17'h1FFFF)
// timeout after 2^17 bits
// the following definition causes shorter timeouts in test mode
`define MINTEST (`MAXLOCK - 17'd260)
// timeout after 260 bits
always @(posedge rxclk)
      if ( !enable || tpat )
         locktime = #1 ( testmode ) ? `MINTEST : 17'h0;     // reset
      else if ( locktime < `MAXLOCK )
         locktime = #1 locktime + 17'h1;     // increment
      else
         locktime = #1 `MAXLOCK;    // force expired
//
always @(posedge rxclk)
      if ( enable && tpat )
         locked = #1 1'b1;    // set
      else if ( !enable || ( locktime == `MAXLOCK ) )
```

```
         locked = #1 1'b0;    // reset
      else
         locked = #1 locked;     // remember
//
endmodule
```

13.3 Appendix: Pool status and Pool Threshold Performance Considerations

It is not a trivial task to estimate a universally applicable required bandwidth on the status channel, as plausible worst-case scenarios are likely to be application-specific, and in any case, dependent on other implementation details such as FIFO depth. However, some general statements can be made without delving too deeply into specifics of particular implementations.

The status channel operates at the serial rate of the data path. For the case of very short packets sent in 32-byte bursts, there is roughly one status update opportunity for each 2-byte data path word. The overhead in the status channel (from parity and framing) can be made sufficiently small with a suitably long Calendar multiple.

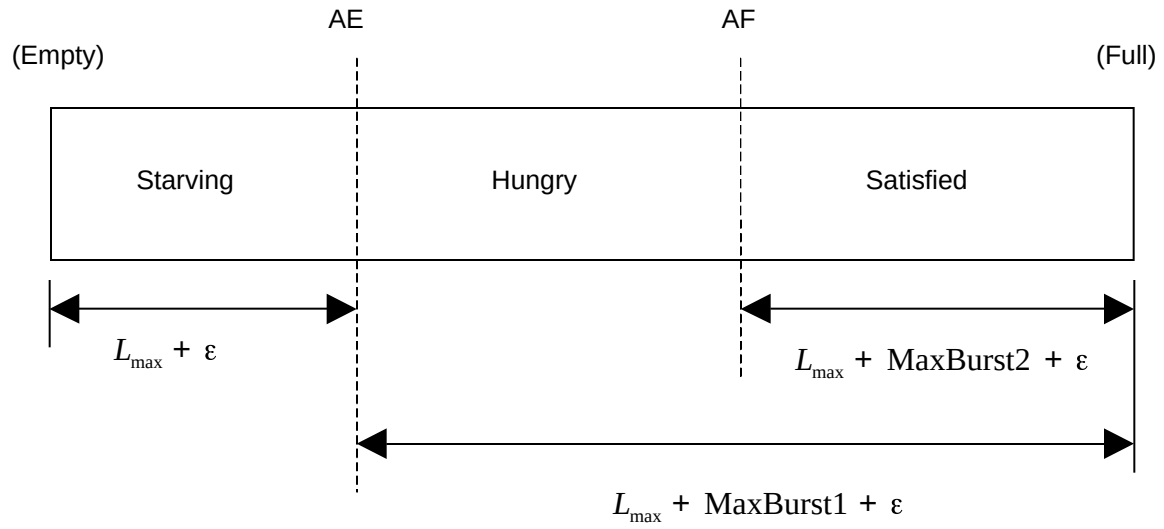
In the more typical cases corresponding to Internet traffic traces, the shortest packets will be roughly 40 bytes long (requiring 21 words to transfer, including control overhead). For a 65-byte packet segmented into two transfers, an average of 35 words will have elapsed between those transfers (33 words for payload and two for control). Hence, such transfers are likely to be in the range of 17.5 to 21 words, ample time for several status update opportunities within the transfer interval. This result does not fundamentally change in multi-port configurations, even with widely different line rates, as each constituent line rate will ultimately bound the amount of data path bandwidth allocated to a given port in the long run. For example, a port running at only a tenth of the total bandwidth supported by the interface can only send and receive data by that same fraction of bandwidth in the long run (excluding transient intervals). By weighting the Calendar such that this port has an average of one update out of every ten opportunities, the corresponding status channel bandwidth for this port is scaled accordingly.

The thresholds for STARVING and HUNGRY are set such that the FIFO can accept at least MaxBurst1 and MaxBurst2 (32-byte) blocks respectively, plus an additional amount to account for feedback-response delay. In order to guard against potential buffer underflow, the lowest threshold must be set high enough to allow the other end to respond to transitions to the state of lowest FIFO occupancy in a reasonable length of time (to the order of the status update interval, plus scheduler response time). MaxBurst2 and MaxBurst1 (if applicable)

must be provisioned to allow adequate utilization of transfer bandwidth between status updates for the given port.

Figure 13.2 shows one possible way for relating FIFO thresholds to MaxBurst1 and MaxBurst2 as well as the response latency. L_{max} corresponds to the worst-case response time, from the delay in receiving a status update over the FIFO status channel, until observing the reaction to that update on the corresponding data path. The quantity ε corresponds to the difference between the granted credit and the actual data transfer length. This difference arises from various protocol overheads and quantization errors in the packet scheduler and the data path.

Figure 13.2: Sample FIFO Thresholds



13.4 Appendix: Data Path Performance Considerations

This section discusses the minimum bus frequencies required for a number of applications, assuming maximum 64-byte payload data transfers. This analysis is done at OC-192 and will translate well to OC-768. The numerical results can be used to provide guidance on the actual operating frequency of the bus. To provide design margin, the actual operating frequency would be higher to account for training and other overhead. Calculations in this section assume that the training bandwidth is insignificant and not included. For the sake of brevity, the word SONET is used to refer to either SONET or SDH.

Since the PHY device may remove or insert framing overhead to data received from or transmitted to the line side, the actual data rate over the SPI interface may be less than the operating line rate. For a given line rate, the actual packet data rate becomes smaller with the packet length, since the framing overhead becomes a larger proportion of the line bandwidth. The SPI

interface, however, inserts a 16-bit control word between payload burst transfers. Hence, the bus tends to operate less efficiently with shorter packet lengths (and therefore tends to require a proportionally higher bus bandwidth for a given data transfer rate). The minimum operating frequency of the bus is the result of an interplay between the control word and the line overhead.

The following cases are considered in this section:
1. ATM cells over SONET.
2. HDLC-framed Packet over SONET.
3. Ethernet LAN PHY Framing.

In case 1, ATM cells are transported back-to-back in the payload area of a SONET frame. There is an 8-bit HEC field in the ATM cell header, which may or may not be transferred over the SPI interface. In case 2, back-to-back HDLC-framed packets are transported in the SONET payload area. Apart from inter-frame flags, the HDLC framing overhead is assumed to consist of an 8-bit Type field, an 8-bit Address field, and a 32-bit frame check sequence (FCS). Byte-stuffing events will increase the line overhead, but these contributions are ignored since the worst-case in this discussion corresponds to the absence of byte stuffing. For cases 1 and 2, the SONET overhead is assumed to be 3.7% of the line rate. In case 3, the packets are encapsulated in Ethernet frames. In this case, jumbo frames (to the order of 9600 bytes long) are also considered, as these frames may be encountered in some applications even though they are longer than the maximum specified by the IEEE.

In general, the (per line) data path rate of the interface, f_I, can be expressed as

$$f_I = f_S \, ABC / W,$$

where

f_S = line rate (9.953 Gb/s for SONET, 10 Gb/s for Ethernet 10 Gb/s LAN PHY).

A = line (Layer 1) efficiency,

$$= \begin{cases} 0.963, & \text{SONET/SDH} \\ 1, & \text{10 Gb/s Ethernet LAN PHY} \end{cases}$$

B = Layer 2 framing efficiency

$= L / L'$, where

$$L' = \begin{cases} L, & \text{53 - byte ATM cells} \\ L+1, & \text{52 - byte ATM cells, POS incl HDLC header/FCS} \\ L+7, & \text{POS excl HDLC header/FCS} \\ L+12, & \text{10 Gb/s Ethernet LAN PHY} \end{cases}$$

C = packet interface overhead

$= (L + (\lceil L/W \rceil - L/W) \times W) + \lceil L/M \rceil \times N) / L$

L = packet length, including packet overhead transferred over interface,

W = interface width (number of parallel data lines),

M = maximum burst transfer size,

N = additional interface overhead per transfer.

$\lceil x \rceil$ = ceiling function (the smallest integer not less than x).

The first, second and third terms in the equation for C correspond to contributions, respectively, from the packet itself, the unused portion (if any) on the last word upon EOP, and the overhead from segmentation. It is assumed in the formula for C that M (in units of bits) is an integer multiple of W. Setting W = 16 bits, M = 64 bytes, N = 2 bytes, the minimum bus frequency is obtained by finding the maximum of f_I over the applicable range of L.

For case 1, the minimum data path line rate, $f_{\min}$, for 52-byte cell transfer on average is 610.37 Mb/s. The corresponding $f_{\min}$ for 53-byte cells on average is 632.97 Mb/s. The peak data path line rates (between SONET framing overhead) correspond to 633.82 Mb/s and 657.29 Mb/s for 52- and 53-byte cells respectively.

For case 2, considering HDLC header and FCS transfer over the interface, the maximum occurs at L = 65 bytes, where the corresponding $f_{\min}$ is on average 635.37 Mb/s. In the more common situation where the header and FCS are not transferred over the interface, $f_{\min}$ increases with L (though not monotonically), approaching an asymptotic limit of roughly 618 Mb/s on average. The corresponding peak data path line rates are 659.78 Mb/s (header and FCS also transferred) and 642 Mb/s (header and FCS not transferred).

Due to the relatively long inter-frame gap (12 bytes) for case 3, the required f_I is fairly small for short L but increases with L as the gap becomes a smaller proportion of the frame length. Hence, the worst case corresponds to jumbo frame transfers. These frames are much longer than the maximum specified by IEEE, but have been used in some applications. An upper bound to $f_{\min}$ can be obtained by assuming that the overhead due to the preamble and the inter-frame gap is negligible for very long frames. Hence, $L' \approx L$ and $f_{\min} \leq$ 644.53 Mb/s.

Table 13.2: Summary of Minimum Data Path Bandwidth Requirements

Traffic Type	SPI-4	SPI-5
ATM Cells, 52 bytes, no HEC	610.37 MHz	2.441 GHz

ATM Cells, 53 bytes, with HEC	632.97 MHz	2.532 GHz
HDLC, 65 bytes, with HDR, CRC	635.37 MHz	2.541 GHz
HDLC, 65 bytes, no HDR, CRC	618.00 MHz	2.472 GHz
Ethernet LAN, 9600 bytes	644.53 MHz	2.578 GHz

14 REFERENCES

Optical Internetworking Forum, SxI-5: Electrical Characteristics of Common I/O for SFI-5 and SPI-5, (To be completed upon approval)

15 GLOSSARY

ACW Address Control Word
ADW Address Data Word
ATM Asynchronous Transfer Mode
BPSCW Bus Parameter Switch Control Word
FCS Frame Check Sequence
ICW Idle Control Word
IEEE Institute of Electrical and Electronics Engineers
LAN Local Area Network
LFSR Linear Feedback Shift Register
LODS Loss of Data Synchronization
LOS Loss of Synchronization
LSB Least Significant Bit
MSB Most Significant Bit
PCW Payload Control Word
PDU Protocol Data Units
PDW Payload Data Word
PHY Physical Layer Device
POS Packet-Over-SONET/SDH
SDH Synchronous Digital Hierarchy
SerDes Serializer/Deserializer
SFI SerDes Framer Interface
SONET Synchronous Optical Network
SPI System Packet Interface
TCW Training Control Word
TDW Training Data Words
UDF User Defined Field
Wcycle Word Cycle

16 APPENDIX: LIST OF COMPANIES BELONGING TO OIF WHEN DOCUMENT IS APPROVED

Accelerant Networks
Accelight Networks
Acorn Networks
Actel
Acterna
ADC Telecommunications
Aeluros
Aerie Networks
Agilent Technologies
Agility Communications
Alcatel
Alidian Networks
All Optical Networks, Inc.
Allegro Networks
Alphion
Altera
Alvesta Corporation
AMCC
America Online
Ample Communications
ANDO Corporation
Anritsu
AON Networks
Appian Communications
Applied Innovation
Applied Optoelectronics
Aralight
Astral Point Communications
AT&T
Atoga Systems
Atrica Inc.
Avici Systems
Axiowave Networks
Bandwidth9
Bay Microsystems
BellSouth Telecommunications
Big Bear Networks
Bit Blitz Communications
Bitmath
Blaze Network Products
Blue Leaf
Blue Sky Research

Bravara
BrightLink Networks
Broadcom
BT
C Speed Corp.
Cable & Wireless
Calient Networks
Calix Networks
Caspian Networks
Catamaran Communications
Celion Networks
Cenix
Centerpoint Broadband Technologies
Centillium Communications
Ceyba
Chiaro Networks
Chunghwa Telecom Labs
Cielo Communications
Ciena Communications
Cinta Corporation
CIR
Cisco Systems
CIVCOM
Clearwater Networks
Coherent Telecom
Computer & Communications Research Labs
Conexant
Continuum Networks
CoreOptics
Coriolis Networks
Corning Incorporated
Corona Optical Systems
Cortina Systems
CORVIS Corporation
CPlane
Crescent Networks
CyOptics
Cypress Semiconductor
Data Connection
Department of Defense
Ditech
Dorsal Networks
Dowslake Microsystems
E2O Communications
EBONE
Efficient Channel Coding

Elisa Communications
Emcore
Emperative
Entridia
Ericsson
ETRI
Extreme Networks
Ezchip
FCI
Fiberhome Telecommunications
Finisar Corporation
FirstWave Intelligent
Flextronics Semiconductor
Force 10 Networks
Foundry Networks
France Telecom
Free Electron Technology
Fujikura
Fujitsu
Furukawa Electric Technologies
Galazar Networks
Gazillion Bits
Gemfire
General Dynamics
Gennum Corporation
Genoa
Geyser Networks
GigaTera
Glimmerglass Networks
Global Crossing North American Networks
Gore & Associates
Gtran
GWS Photonics
Harting Electro-Optics GmbH
Helic S.A.
Helix AG
Hi/fn
Hitachi
Honeywell
Huawei Technologies
Hyperchip
IBM Corporation
Ignis Optics
Infineon Technologies
Innovance Networks
Inphi

Integrated Device Technology
Intel
Internet Machines
Interoute Telecom Germany
Intune Technologies, Ltd.
Iolon
Japan Telecom
Jedai Broadband Networks
Jennic
Juniper Networks
KDDI R&D Laboratories
KereniX
Kestrel Solutions
Kirana Networks
Kodeos Communications
Korea Telecom
Lambda Crossing
Laurel Networks
Lightbit Corporation
LSI Logic
Lucent
Lumentis
LuxN
LYNX - Photonic Networks
Mahi Networks
Maple Optical Systems
Marconi Communications
Maxim Integrated Products
Memlink
Meriton
Metro-OptiX
MindTree Consulting Pvt. Ltd
Mintera
Mitsubishi Electric Corporation
Movaz Networks, Inc.
Multilink Technology Corporation
Myrica Networks
National Semiconductor
Nayna Networks
NEC
Net Brahma Technologies
Network Associates
Network Elements
Network Photonics
New Focus
NIST

Nokia
Nortel Networks
NTT Corporation
NurLogic Design
Ocular Networks
OKI Electric Industry
Onex Communications
ONI Systems
OpNext
Opthos
Optical Datacom
Optical Switch
Optillion
Optivera
Optix Networks
Optobahn
OptronX
PacketLight Networks
Paracer
Parama Networks
Paxonet Communications
Peregrine Semiconductor
Peta Switch Solutions
Philips Semiconductors
Photonami, Inc.
PhotonEx
Photuris, Inc.
Phyworks
PicoLight
Pine Photonics Communications
Pluris
PMC Sierra
Polaris Networks, Inc.
Power X Networks
Procket Networks
Quake Technologies
Quantum Bridge
Qwest Communications
Radiant Photonics, Inc.
Raza Foundries
Redback Networks
RedClover Networks
Redfern Broadband Networks
RF Micro Devices
Riverstone Networks
Sandia National Laboratories

Santur
Siemens
Sierra Monolithics
Silicon Access
Silicon Bridge
Silicon Labs
Silicon Logic Engineering
Silicon Packets, Inc.
SiPackets, Inc.
SITA
Solidum
Sorrento Networks
Sparkolor
Spirent Communications
Sprint
StrataLight Communications
Sumitomo Electric Industries
Sun Microsystems
Sycamore Networks
Syntera Communications
TDK Semiconductor
Tektronix
Telcordia Technologies
Tellabs
Tellium
TelOptica
Tenor Networks
TeraBeam Networks
TeraBurst Networks
Teradiant Networks, Inc.
Terago Communications
Texas Instruments
TILAB S.p.A.
T-Networks, Inc.
Toshiba Corporation
Transparent Networks
Trellis Photonics
TriCN Associates, LLC
TriQuint Semiconductor
Tropic Networks Inc.
TRW
T-Systems Nova
Turin Networks
TyCom
US Conec
Valiant Networks

Velio Communications
Verizon
Versanetworks
Village Networks
VIPswitch
Virata Corporation
Vitesse Semiconductor
Vivace Networks
VSK Photonics
Wavium AB
West Bay Semiconductor
White Rock Networks
Williams Communications
Xanoptix
Xelerated Packet Devices
Xilinx
Xindium
Xlight Photonics
XLOptics
Yotta Networks
Zagros Networks
Zarlink Semiconductor
Zepton Networks
ZettaCom